ABRIL 86 - 300 ptas.

TodoSpectrum

AÑO II - Número 20

REVISTA EXCLUSIVA PARA USUARIOS

DESCUBRIMOS

Commando: Guía del hacker

INVESTIGAMOS

Compilación de SuperBASIC

PROBAMOS

Termometro para Spectrum

PROGRAMAMOS

Ficheros secuenciales en Pascal

IMPRIMIMOS

Copy de grises

Una mejor impresión, sin depender solamente de los blancos y negros. Ahora pueden obtenerse gran variedad de grises.

Suplemento QL

LA BATALLA DE LOS CHIPS

68008 versus Z-80

SPECTRUM 128
EL SUMMUM

Spectrum, como líder, marca un nuevo hito en la historia de los ordenadores familiares.

El Spectrum 128.

Gran capacidad de memoria. Teclado y mensajes en castellano, teclado independiente para operaciones numéricas y de tratamiento de textos...

Sinclair e Investrónica han desarrollado una auténtica novedad. En ningún lugar del mundo, salvo en los Distribuidores Exclusivos de Investrónica, podrás encontrar el nuevo Spectrum 128.

Sé el primero en tener lo último.

SPECTRUM 128. NOVISSIMUS



investronica

Tomás Bretón, 62.
Tel. (91) 467 82 10.
Telex 23399 IYCO E.
28045 Madrid

Camp, 80.
Tels. (93) 211 26 58 - 211 27 54.
08022 Barcelona

SUMARIO

AÑO II - Nº 20 - ABRIL 1986





SERVICIO DE EJEMPLARES ATRASADOS



Complete su colección de

Todospectrum

A continuación le resumimos el contenido de los ejemplares aparecidos hasta ahora.

Núm. 2 - 300 ptas.
Gráficos profesionales/Desplazamiento pixel a pixel/Utilización de rutinas/Construcción del interface centronics/Programas de utilidad para microdrive/Rutina reset en código máquina/Análisis del editor de textos Tasword/Interfaces para impresoras/Programas.

Núm. 3 - 300 ptas.
Novedades sonimag'84/Ampliando el Basic/Programas para ordenar programas/Gráficos con el VU-3D/Lenguaje Forth/Archivos en microdrive/Programación de un interface de impresora/Programas.

Núm. 4 - 300 ptas.
De profesión: programador/Consola para el Spectrum/Comparación código máquina-Basic/Análisis programa contabilidad/Calendario/Pascal/Programas.

Núm. 5 - 300 ptas.
Floppys para Spectrum/Diseño asistido por ordenador/64 Caracteres por linea/Juego de la vida/Pascal/Asi hacemos las portadas/Control de evaluaciones/Programas.

Núm. 6 - 300 ptas.
Representación de funciones/Todos los caminos conducen a la ROM/Juegos/Pascal/Construcción de un lápiz óptico/Programas de gestión. El SITI/Logo: torgugas para todos/ Interrupciones del Z-80/Programas.

Núm. 7 - 300 ptas.
Del 48 al PLUS paso a paso/¿Plotter para Spectrum?/Juegos/Libros de código máquina/Lápiz óptico. Programación del montaje/El LOGO en la escuela/Pascal/Floppys para Spectrum/Programas.

Núm. 8 - 300 ptas.
Amplia tu memoria... a 48 K/Arquitectura: análisis del PREYME/Juegos/FORTH. Nociones básicas/Una clave, please/QL Magazine. Ultimas novedades, análisis de software, Lenguajes/Aula informática con Spectrum/Programas.

Núm 9 - 300 ptas.
Spectrum parlanchin/Juegos/Aula informática con Spectrum/Análisis: Comercial 4/Pascal/Periféricos: Wafadrive/QL Magazine: EASEL lo mejor de PSION. Música con QL/Desplazamiento Pixel a Pixel, aportación de lectores/Programas/Programer II.

Núm. 10 - 300 ptas.
Discos: invesdisc 200/Juegos/Dos programas simultáneos/Protección del software/Conozca extremadura, consulte a su ordenador/Desensamblador Z-80/Sofware educativo/QL Magazine: novedades Informat, Hoja de cálculo, Ajedrez/Construya su propio Joystick/Pascal/programas.

DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX
(sin necesidad de encuademación)

Núm. 11 - 300 ptas.
Actualidad/La otra cara del LOGO/Juegos/El Spectrum habla castellano/SOFTAID ayuda para Etiopia/S.O.S. aquí el Spectrum/Dibujar con lápiz óptico/QL Magazine: Procesador de textos. Teclas de función programables/Programas.

Núm. 12 - 300 ptas.
Actualidad/Inteligencia artificial/Lápiz óptico dk'TRONICS/Juegos/Análisis/Bingo/Z-80 PIO/Código máquina/Análisis: MASTERFILE/Programas.

Núm. 13 - 300 ptas.
Actualidad/Discos: Discovery 1/Juegos/Inteligencia artificial/Un nuevo sistema operativo/QL Magazine: Archive, Cartridge doctor. Aplicaciones comerciales/Código máquina/Programas.

Núm. 14 - 300 ptas.
Actualidad, Spectrum 128/Cálculo de estructuras para ingenieros y arquitectos/HELP utilidades en microdrive/Juegos/El microdrive ese desconocido/Código máquina/QL Magazine: GRAPHIC QL. Juegos. Discos de 720 K/Un nuevo operativo/Programas.

Núm. 15 - 300 ptas.
Actualidad/Spectrum 128/Un nuevo operativo/Circulos redondos/Juegos/Utilidades: BETA-BASIC/QL Magazine: Introducción al SUPER BASIC. Nuevas utilidades/Hardware: Puertas lógicas/Código máquina/Programas.

Núm. 16 - 300 ptas.
Actualidad/Cinco horas con SCREEN$/Hardware práctico/Cálculos de infinita precisión/Juegos/Un nuevo operativo/QL Magazine: Gráficos en SUPER-BASIC. Dibujando con ratón. Archivos con Archive. Programa/La última batalla, Juego estratégico.

Núm. 17 - 300 ptas.
Actualidad/Gráficos interactivos/Juegos/Código máquina/Un nuevo operativo/Trucos de programación/QL Magazine: Radiografía del QL. Gráficos en SUPER-BASIC/Libros/Programmas.

Núm. 18 - 300 ptas.
Actualidad/Introducción al C/Libros/Juegos/De cinta a microcinta/Visión panorámica de los microprocesadores más comunes/QL Magazine: Copy de grises. Microprocesadores 68000, una familia numerosa/Curioseando en la ROM/Programas.

Para hacer su pedido, rellene este cupón HOY MISMO y envíelo a:

Todospectrum Bravo Murillo, 377
Tel. 733 96 62 - 28020 MADRID

Ruego me envien los siguientes ejemplares atrasados de TODOSPECTRUM .... al precio de 300 pts.

El importe lo abonaré
☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta ....

Fecha de caducidad .... Firma

NOMBRE ....

DIRECCION ....

CIUDAD .... C P ....

PROVINCIA ....

DIRECTOR:
Enrique F. Larreta
REDACTOR JEFE:
Emiliano Juárez
REDACCION:
Ignacio Borrell, Octavio López,
Antonio del Rio
DISEÑO:
Ricardo Segura y Benito Gil

Editado por PUBLINFORMATICA, S. A.
Bravo Murillo, 377. 5.º A. Tel.:
733 74 13 - 28020 Madrid
Presidente:
Fernando Bolin
Director Editorial Revistas de Usuarios:
Juan Arencibia
Director de Ventas:
Antonio González
Producción: Miguel Onieva
Servicio al cliente:
Julia González. Tel.: 733 79 69
Administración:
PUBLINFORMATICA, S. A.
Publicidad Madrid:
Emilio García
Dirección, Publicidad y Administración:
Bravo Murillo, 377. 5.º A. Tel. 733 74 13.
Télex: 48877 OPZX e 28020 Madrid
Publicidad Barcelona:
Lidia Cendrós. Pelayo, 12. Tels. (93)
318 02 89 - 301 47 00, ext. 27 y 28.
08001 Barcelona
Depósito legal: M-29041-1984
Distribuye S.G.E.L. Avda. Valdelaparra,
s/n. Alcobendas (Madrid).
Fotomecánica: Karmat, C/ Pantoja, 10.
Madrid.
Fotocomposición: Artecomp.
Imprime: Héroes, C/ Torrelara, 8. Madrid.
Distribuidor en VENEZUELA,
SIPAM, S. A.
AVD. REPUBLICA DOMINICANA, EDIF.
FELTREC - OFICINA 4B BOLEITA SUR
CARACAS (VENEZUELA)
Esta publicación es miembro de la
Asociación de Revistas de
Información ari asociada a la
Federación Internacional de Prensa
Periódica, FIPP.

SUSCRIPCIONES:
Rogamos dirijan toda la correspondencia relacionada con suscripciones a:
TODOSPECTRUM EDISA: Tel. 415 97 12
C/ López de Hoyos, 141-5º
28002 MADRID
(Para todos los pagos reseñar solamentes TODOSPECTRUM)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
TODOSPECTRUM
C/ Bravo Murillo, 377. 5.º A
Tel. 733 74 13 - 28020 MADRID

Si deseas colaborar en TODOSPECTRUM remite tus artículos o programas a Bravo Murillo, 377. 5.º A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados. A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

EDITORIAL

LA PUNTA DEL ICEBERG



El mes pasado, gracias a una espectacular acción policial en el Rastro madrileño, la piratería saltó al primer plano de la actualidad. Veintiocho personas fueron retenidas transitoriamente y unas once mil copias piratas de programas (la mayoría para Spectrum) quedaron en poder de la policía.

No obstante, conviene recordar que el Rastro es únicamente la punta de un enorme iceberg que proporciona pingües ganancias a unos cuantos desaprensivos. Los vendedores del Rastro saldrían bien parados si los comparásemos, por ejemplo, con los de algunos comercios especializados en los que no sólo se pueden conseguir programas originales, sino también, a poco que insistamos y a precios inferiores, copias piratas de esos mismos programas.

Tampoco hay que olvidar que sin la posibilidad de obtener fácilmente copias de cualquier juego, el Spectrum no sería el ordenador más difundido y popular en España. Hace casi exactamente tres años, Arturo Selgas, entonces director de marketing de Investrónica, afirmaba al respecto: «No protegeremos los programas de copia ni de listado. El objetivo de un ordenador de este tipo es que la gente lo analice, aprenda a usarlo y le saque aplicaciones que, tal vez, fueran insospechadas para el diseñador».

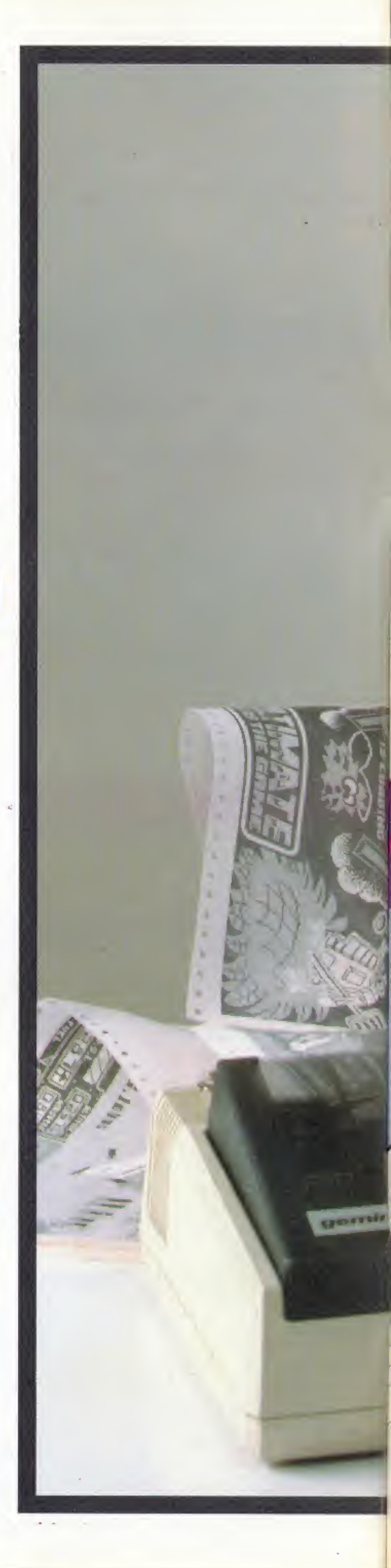
Viene todo esto a cuento del excesivo revuelo provocado por los sucesos del Rastro madrileño. Con medidas de este tipo difícilmente se eliminará la piratería. Mientras el software sea inasequible para la mayoría de los compradores potenciales debido a su elevado precio, esta lacra continuará existiendo, aunque quizás más subrepticiamente.

TODOSPECTRUM

SE ACABO EL BLANCO Y NEGRO

COPY DE GRISES

Muchas veces se habrá sentido defraudado al hacer una copia de pantalla en su impresora matricial. Aunque su Interface esté provisto de una rutina de volcado en alta resolución, la imagen resultante es de una calidad pésima en relación con la original de la pantalla del monitor. Esto se debe a que interpreta la tinta como negro y el papel como blanco. La rutina que le proponemos va más allá; permite hacer copias de pantalla distinguiendo los colores originales para transformarlos en sus respectivos tonos grises equivalentes, de tal forma que la imagen resultante en la impresora es la misma que vemos en un televisor en blanco y negro.



ESTA rutina permite hacer una copia o volcado por impreso (Epson RX80, FX80 y compatibles) con todos los matices de sombreado que pueden apreciarse en el televisor en blanco y negro conectado a su Spectrum. No solamente hace esto (que ya es mucho), sino que produce una copia impresa realizada en sentido vertical (28 cm + 15 cm), con lo cual se aprovechan las máximas dimensiones del papel y se evitan las deformaciones de las figuras (imágenes alargadas).

El programa cargador en Basic

El cargador BASIC de la figura 1 funciona directamente para utilizarlo con un interface Hilderbay. En caso de que su interface sea el Kempston E sólo debe hacer unas pequeñas modificaciones en el programa, sustituyendo las DATAS de las líneas 1.000 y 1.300 de la figura 1 por los de la figura 2. Si el interface que posee no es ninguno de los anteriores, no tendrá más



star

FIGURA 1
Cargador para Interface Hilderbac

```
 100 CLEAR 65117
 110 LET n=0: FOR i=65118 TO 653
07: READ a: POKE i,a: LET n=n+PE
EK i: NEXT i
 120 IF n<>16665 THEN  PRINT "ER
ROR EN DATAS": STOP
 130 PRINT "CODIGO CORRECTO": ST
OP
1000 DATA 205,0,91,24,15,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0
1100 DATA 62,27,205,20,255,62,65
,205,20
1110 DATA 255,62,3,205,20,255,14
,0,62
1120 DATA 27,205,20,255,62,42,20
5,20,255
1130 DATA 62,4,205,20,255,62,16,
205,20
1140 DATA 255,62,2,205,20,255,6,
0,197
1150 DATA 205,170,34,71,4,62,1,1
5,16
1160 DATA 253,166,8,124,15,15,15
,230,3
1170 DATA 246,88,103,70,8,120,32
,3,15
1180 DATA 15,15,230,7,33,244,254
,135,135
1190 DATA 95,22,0,25,6,3,126,205
,20
1200 DATA 255,35,16,249,193,4,12
0,254,176
1210 DATA 56,199,62,13,205,20,25
5,62,10
1220 DATA 0,0,0,12,32,159,62,27,
205
1230 DATA 20,255,62,65,205,20,12
7,62,12
1240 DATA 205,20,255,201,224,224
,224,0,192
1250 DATA 96,192,0,160,64,160,0,
32,64
1260 DATA 128,0,96,0,96,0,64,0,6
4
1270 DATA 0,0,64,0,0,0,0,0,0
1300 DATA 197,229,205,254,91,225
,193,201,-1
```

FIGURA 2
Modificaciones para Interface Kempston

```
1000 DATA 62,3,205,1,22,24,13,0,
0,0,0,0,0,0,0,0,0,0,0,0
1300 DATA 245,62,27,215,241,215,
201
```

remedio que modificar el código máquina.

En el supuesto de que su impresora necesite avances de línea después del retorno de carro, debe insertar en el cargador la siguiente línea:
230 POKE 32478,205 : POKE 32479,20 : POKE 32480,127

La rutina funciona tanto en el Spectrum de 16 K como en el de 48 K, ya que se ejecuta a partir de la dirección 32350. Si su aparato es de 48 K y desea colocar el código en una dirección más alta (65118), debe cambiar todas las secuencias de Datas '205, 20, 127' por '205, 20, 255' y la secuencia '33, 244, 126' por '33, 244, 254'.

Ejecución de la rutina

Una vez haya hecho funcionar el programa cargador y no se presente ningún error (en tal caso deberá revisar la secuencia de datas), puede grabar la rutina con SAVE «COPYGRISES» CODE 32350,250 (o 65118,250 si ha decidido colocar la rutina en una dirección más alta).

Para comprobar su perfecto funcionamiento, cargue una pantalla cualquiera (preferentemente con muchos colores) mediante SCREEN$ "" y acto seguido ejecute la rutina mediante RANDOMIZE USR 32350 (o RANDOMIZE USR 65118 para el código en una dirección más alta). Si la impresora empieza a imprimir, es muy probable que no haya errores, pero si se produce un *crash* del sistema, vuelva a cargar el listado BASIC y compruebe de nuevo los códigos.

Si interrumpe la ejecución de la rutina durante la impresión, la impresora actuará de un modo extraño, con lo cual el mejor medio para interrumpir el volcado consiste en desconectar la impresora y empezar de nuevo.

Fundamentos

Los sombreados de la copia, simulando los colores de la pantalla, se realizan aprovechando la mayor resolución de la impresora. Teniendo en cuenta la resolución de 176 × 256 pixels del Spectrum, si cada uno de estos pixels estuviera representado por una matriz de 3 ×



El copy con la rutina incluida en el interface Kempston carece de matices grises.

3 puntos en la impresora, obtendríamos una copia con una resolución de 528 × 728. Por esta razón debemos descartar las antiguas Epson (tipo MX) con una resolución de sólo 480 puntos por línea. Las Epson actuales tienen una resolución de 640 puntos por línea en modo 4, con lo cual son perfectamente válidas para nuestros propósitos. La distancia del avance de línea vertical debe ser de 3 puntos (3/72 pulgadas), lo cual se consigue mediante el comando ESC "A".

Para hacerse una idea de cómo se consiguen los sombreados, la figura 3 muestra las matrices de 3 puntos utilizadas para este fin. Por supuesto, usted es libre de cambiar a su gusto la rejilla de puntos, para conseguir otros tonos diferentes de grises, incluso puede invertir la secuencia de sombreado, de tal forma que cuando aparezca en la pantalla un bit de tinta blanca, se imprima en el papel una matriz de 3 × 3 puntos, y así sucesivamente, consiguiéndose una copia en «negativo» respecto al original de la pantalla.

Esta rutina permite una copia por impresora con todos los matices de sombras apreciables en un televisor en B/N

El listado ASSEMBLER

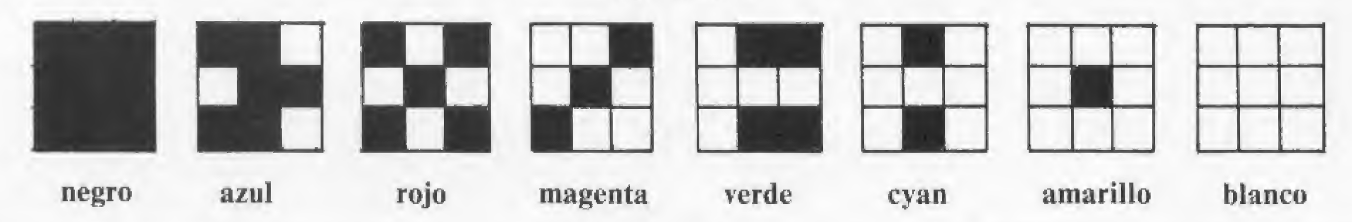
El listado fuente en ensamblador de la figura 4 está preparado para que una vez ensamblado funcione directamente con el interface Hilderbay. Para el Kempston E, sustituya las instrucciones de las subrutinas COPY y ITFCE por las de la figura 5. Si su interface no corresponde a ninguno de estos dos, puede operar de la misma manera, creando previamente las instrucciones necesarias.

La rutina propiamente dicha comienza en la etiqueta BEGIN, donde se establece el valor de la distancia del avance de línea a 3/72 pulgadas y se inicializa el registro C a cero (utilizado como contador de la coordenada X). Seguidamente, se sitúa la impresora

FIGURA 4
Listado fuente en ensamblador.

```
10          ORG   32350
20          ENT   $
30          CALL  23296
40          JR    COPY
50          END
55          ORG   32370
60  COPY    LD    A,27          ;ESC
70          CALL  ITFCE
80          LD    A,"A"
90          CALL  ITFCE
100         LD    A,3
110         CALL  ITFCE
120         LD    C,0
130 LINEA   LD    A,27          ;ESC
140         CALL  ITFCE
150         LD    A,"*"
160         CALL  ITFCE
170         LD    A,4           ;MODO 4
180         CALL  ITFCE
190         LD    A,16
200         CALL  ITFCE
210         LD    A,2
220         CALL  ITFCE
230         LD    B,0
240 NXY     PUSH  BC
250         CALL  )22AA
260         LD    B,A
270         INC   B
280         LD    A,1
290 BUC     RRCA
300         DJNZ  BUC
310         AND   (HL)
320         EX    AF,AF
330         LD    A,H
340         RRCA
350         RRCA
360         RRCA
370         AND   3
380         OR    )58
390         LD    H,A
400         LD    B,(HL)
410         EX    AF,AF'
420         LD    A,B
430         JR    NZ,TINTA
440         RRCA
450         RRCA
460         RRCA
470 TINTA   AND   7
480         LD    HL,TABLA
490         ADD   A,A
500         ADD   A,A
510         LD    E,A
520         LD    D,0
530         ADD   HL,DE
540         LD    B,3
550 OUTLP   LD    A,(HL)
560         CALL  ITFCE
570         INC   HL
580         DJNZ  OUTLP
590         POP   BC
600         INC   B
610         LD    A,B
620         CP    176
630         JR    C,NXY
640         LD    A,13
650         CALL  ITFCE
660         LD    A,10
670         NOP
680         NOP
690         NOP
700         INC   C
710         JR    NZ,LINEA
720         LD    A,27          ;ESC
730         CALL  ITFCE
740         LD    A,"A"
750         CALL  ITFCE
760         LD    A,12
770         CALL  ITFCE
780         RET
790 TABLA   DEFB  %11100000     ;NEGRO
800         DEFB  %11100000
810         DEFB  %11100000
820         DEFB  0
830         DEFB  %11000000     ;AZUL
840         DEFB  %01100000
850         DEFB  %11000000
860         DEFB  0
870         DEFB  %10100000     ;ROJO
880         DEFB  %01000000
890         DEFB  %10100000
900         DEFB  0
910         DEFB  %00100000     ;MAGENTA
920         DEFB  %01000000
930         DEFB  %10000000
940         DEFB  0
950         DEFB  %01100000     ;VERDE
960         DEFB  %00000000
970         DEFB  %01100000
980         DEFB  0
990         DEFB  %01000000     ;CYAN
```

FIGURA 3



Tramas utilizadas en la rutina.

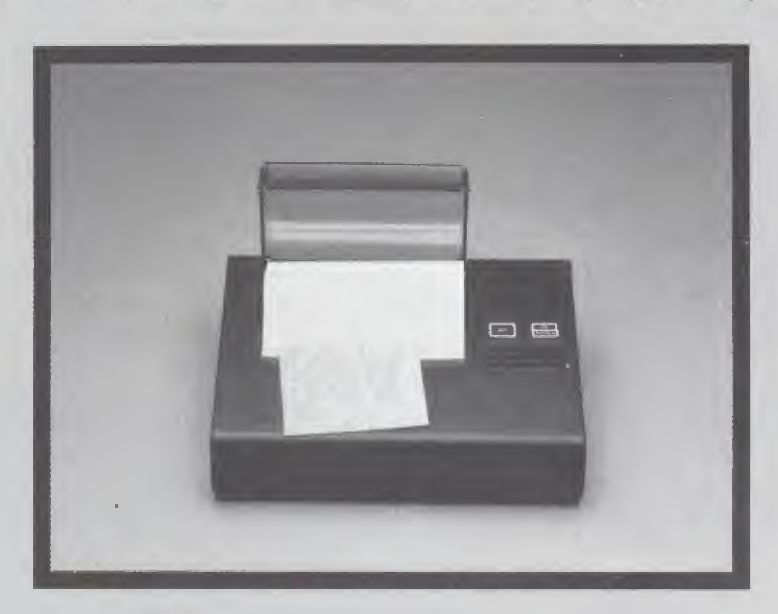
```
1000            DEFB %00000000
1010            DEFB %01000000
1020            DEFB 0
1030            DEFB %00000000  ;AMA
RILLO
1040            DEFB %01000000
1050            DEFB %00000000
1060            DEFB 0
1070            DEFB 0          ;BLA
NCO
1080            DEFB 0
1090            DEFB 0
1100            DEFB 0
1110            END
1120 ITFCE      PUSH BC
1130            PUSH HL
1140            CALL 23550
1150            POP  HL
1160            POP  BC
1170            RET
```

en modo 4 y se carga 0 en el registro B (contador de la coordenada Y). A continuación se hace una llamada a la rutina ROM 22AA, que calcula la posición de pantalla de HL a partir de las coordenadas de pantalla de B y C, así como el color del pixel (a partir del archivo de atributos). Se lee la TABLA y se imprimen los puntos correspondientes (repitiéndose esto para las 176 posiciones de Y). Después de esto, se pasa a una nueva línea (y opcionalmente a un avance de línea) para imprimir todas las 256 posiciones de X. Finalmente, se inicializa el avance de línea a 12/72 pulgadas.

Los datos de TABLA contienen los diseños para cada color, y no supone ningún problema cambiarlos para conseguir diferentes efectos.

Orlando Araujo Martín

COPY DE GRISES CON LA ZX-PRINTER



CUANDO se realiza un dibujo con el Spectrum, normalmente se intentan aprovechar sus posibilidades de color al máximo, pudiéndose obtener pantallas bastante espectaculares. Pero a la hora de realizar una copia en impresora de cualquier pantalla, el ordenador no entiende de colores: se fija sólo en si un determinado *pixel* es de papel o de tinta, y no en cuáles son esos colores de papel y tinta. Por esta causa, algunas pantallas pueden salir muy mal. Para conseguir que queden bien es necesario tener en cuenta los colores, sustituyéndolos por distintas tonalidades de grises. Es algo así como pasar el dibujo de color a blanco y negro.

Las tramas

Para conseguir los distintos grises se utilizan una serie de tramas más claras o más oscuras, haciendo corresponder a cada color con una trama que tenga más o menos su mismo nivel de brillo o de claridad. En impresoras más grandes, que tienen mayor resolución de puntos, se puede hacer corresponder un *pixel* de la pantalla del Spectrum con un rectángulo de puntos en el papel de la impresora. Se puede utilizar entonces este rectángulo para conseguir definir las diversas tramas. Pero esto no puede hacerse en las impresoras tales como la ZX Printer o la GP50S, a las que va dedicado este artículo, ya que tienen una resolución idéntica a la del ordenador. En este caso es necesario modificar un poco el sistema, pues con un solo punto en la impresora por cada punto del ordenador, no podemos definir ninguna trama. Será necesario, por tanto, que las tramas ocupen más de un *pixel* de pantalla. En los programas que acompañan a este artículo se han elegido como dimensiones de las tramas las de un cuadrado de cuatro por cuatro *pixels*. Son las dimensiones idóneas para hacer el programa lo más corto y sencillo posible y para optimizar los resultados obtenidos. Las tramas utilizadas se pueden ver en la figura 1.

El programa realiza primero la conversión a gris en la pantalla. Luego la copia como lo haría normalmente, pero incluyendo las dos líneas inferiores

El programa

El listado 1 realiza un COPY de grises utilizando las tramas anteriormente citadas. Primero realiza la conversión a gris dentro de la propia pantalla y luego la copia como se haría normalmente, sólo que incluyendo las dos líneas inferiores.

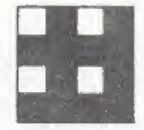
La razón de realizar esta transformación sobre la pantalla es porque simplifica el programa y lo hace más fácil de entender y modificar.

El funcionamiento del programa es el siguiente: Primero inicializa los registros DE y HL con los valores de las direcciones de comienzo de pantalla y atributos respectivamente. Tras esto se entra en un bucle que se repetirá para cada uno de los 768 caracteres de la pantalla

FIGURA 1


NEGRO

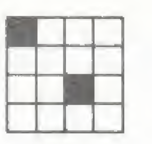

AZUL


ROJO


MORADO

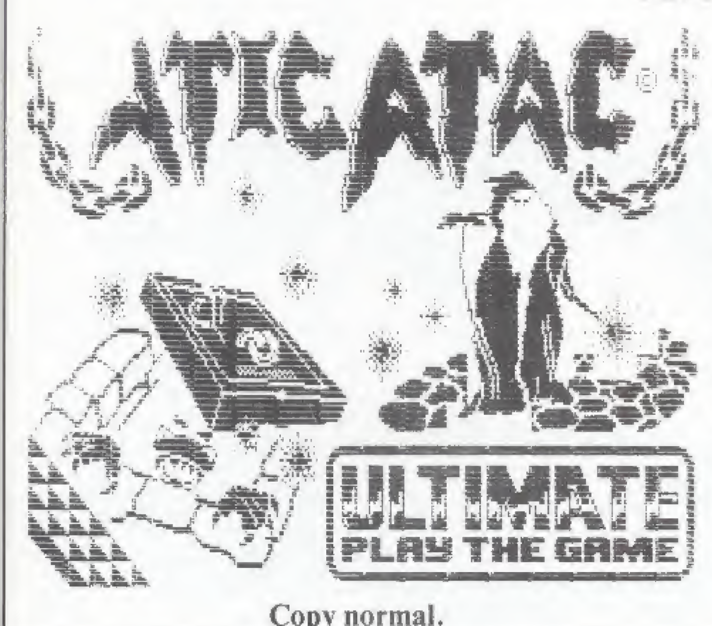

VERDE


CYAN


AMARILLO


BLANCO

FIGURA 2



Copy normal.



Con rutina de grises.

(24 x 32) en orden de izquierda a derecha y de arriba a abajo. Dentro de este bucle se toman los valores de papel y de tinta multiplicados por ocho para buscar la trama correspondiente en la tabla de tramas que hay al final del listado. En el listado están definidas con las dimensiones de un carácter para simplificar cálculos, pero en realidad se repiten cada cuatro *pixels*, tanto en horizontal como en vertical. Se deja en BC la dirección de la trama del papel, y en HL la de la tinta. Entonces se entra en un bucle que se repite ocho veces para los ocho *bytes* que componen un carácter. Dentro de este bucle se utilizan las tramas como máscaras para los *pixels* del *byte* mediante la operación lógica AND, y se deja el resultado nuevamente en la dirección que corresponda. Antes de entrar en dicho bucle, el *byte* de atributos se pone a papel blanco y tinta negra. Después de transformar todo el carácter se actualizan las direcciones de pantalla y atributos y se cierra el bucle comprobando si la dirección señalada por DE sigue siendo del archivo de pantalla. Cuando no es así se da por terminada la transformación y se procede a la copia en impresora mediante la rutina de la ROM, pero cargando antes el valor 192 en B para que no se «coma» las dos últimas líneas.

El factor más importante para obtener buenos resultados con ambas rutinas es la configuración de las tramas. Siempre es posible encontrar alguna que se adapte mejor al color correspondiente

Puede introducir el programa con el listado ensamblador o con el cargador BASIC. Aunque no tiene llamadas ni saltos absolutos, no es reubicable porque utiliza una dirección absoluta: la de la tabla de tramas. Si quiere ubicarlo en otro sitio, tras haberlo grabado en cinta, cárguelo en la dirección deseada y haga: RANDOMIZE dir+80 : POKE DIR+21, PEEK 23670 : POKE DIR+22, PEEK 23671. Después grábelo desde su nueva dirección. La longitud es de 139 *bytes*. Para utilizarlo cárguelo mediante la instrucción LOAD " "CODE dirección. Una vez hecho esto cargaremos la pantalla que deseemos copiar y haremos RANDOMIZE USR dirección, con cuidado de hacer estas dos últimas operaciones de una sola vez si no queremos que se borren las dos últimas líneas.

En la figura 2 se puede contemplar una copia hecha con esta rutina frente a una copia de la misma pantalla hecha sin esta rutina. La diferencia es notable. Los dibujos, en general, suelen quedar bastante bien, pero hay un problema a la hora de copiar letras, pues como las tramas ocupan más de un *pixel*, puede borrarse algún punto de la letra, y, a no ser que estén en blanco y negro, quedarán irreconocibles. Este problema no tiene, en realidad, ninguna solución, aparte de comprar una impresora mejor. Sin embargo, no hay que desanimarse, pues existe un pequeño truco que, aunque tiene un defecto,

Listado 1 A
CAMBIO FUENTE EN ENSAMBLADOR

```
10 ;
20 ;
30 ;   COPY  DE .GRISES
40 ;
50 ;
51 ;POR PABLO ARIZA-1986
52 ;
53 ;
60          ORG   50000
70 ;
80          LD    HL,22528
90          LD    DE,16384
100 LOD     PUSH  DE
110         PUSH  HL
120         LD    A,(HL)
130         AND   7
140         RLCA
150         RLCA
160         RLCA
170         PUSH  AF
180         LD    A,(HL)
190         LD    (HL),56
200         AND   56
210         LD    BC,NEGRO
220         LD    H,0
230         LD    L,A
240         ADD   HL,BC
250         POP   AF
260         PUSH  HL
270         LD    H,0
280         LD    L,A
290         ADD   HL,BC
300         POP   BC
310         LD    A,B
320 LAP     EX    AF,AF'
330         LD    A,(DE)
340         PUSH  DE
350         LD    E,A
360         AND   (HL)
370         LD    D,A
380         LD    A,E
390         CPL
400         LD    E,A
410         LD    A,(BC)
420         AND   E
430         OR    D
440         POP   DE
450         LD    (DE),A
460         INC   D
470         INC   HL
480         INC   BC
490         EX    AF,AF'
500         DEC   A
510         JR    NZ,LAP
520         POP   HL
530         POP   DE
540         INC   HL
550         INC   E
560         LD    A,E
570         AND   A
580         JR    NZ,VU
590         LD    A,D
600         ADD   A,8
610         LD    D,A
620 VU      LD    A,D
630         CP    88
640         JR    C,LOD
650         DI
660         LD    B,192
670         JP    )0EAF
680 ;
690 ;
700 ;  TRAMAS DE LOS COLORES
710 ;
720 ;
730 NEGRO   DEFB 255,255,255,25
5,255,255,255,255
740 ;
750 ;
760 AZUL    DEFB 119,255,221,25
5,119,255,221,255
770 ;
780 ;
790 ROJO    DEFB 85,255,85,255,
85,255,85,255
800 ;
810 ;
820 MORADO DEFB 170,85,170,85,
170,85,170,85
830 ;
840 ;
850 VERDE   DEFB 136,85,34,85,1
36,85,34,85
860 ;
870 ;
880 CYAN    DEFB 170,0,170,0,17
0,0,170,0
890 ;
900 ;
910 AMARI   DEFB 136,0,34,0,136
,0,34,0
920 ;
930 ;
940 BLANCO DEFB 0,0,0,0,0,0,0,
0
```

Listado 1 B
CARGADOR BASIC DEL COPY DE GRISES

```
  1 DATA "210058110040D5E57EE60
7070707F57E3638E63801A0C326006F0
9F1E526006F09C13E08081AD55F0DD6"
  2 DATA "A6577B2F5F0AA3B2D1121
4230308 3D20EBE1D1231C7BA720047AC
608577AFE5838BCF306C0C3AF0E10A5"
  3 DATA "FFFFFFFFFFFFFFFF77FFD
DFF77FFDDFF55FF55FF55FF55FFAA55A
A55AA55AA5588552255885522551A90"
  4 DATA "AA00AA00AA00AA0088002
200880022000000000000000003FC"
 200 CLEAR 39999: RESTORE : LET
a=10: LET b=11: LET c=12: LET d=
13: LET e=14: LET f=15: LET ad=5
e4
 210 PRINT AT 0,0;"LEYENDO LINEA
:": FOR z=1 TO 4: PRINT AT 0,16;
z
 220 READ a$: LET b$=a$(LEN a$-3
 TO ): LET a$=a$( TO LEN a$-4):
IF LEN a$/2<>INT (LEN a$/2) THEN
  PRINT  FLASH 1;AT 0,0;"LINEA I
MPAR EN": STOP
 230 LET w=0: FOR x=1 TO LEN a$
STEP 2: LET v=VAL a$(x)*16+VAL a
$(x+1): LET w=w+v
 240 POKE ad,v: LET ad=ad+1: NEX
T x
 250 LET v=0: FOR x=1 TO 4: LET
v=v*16+VAL b$(x): NEXT x: IF v<>
w THEN  PRINT  FLASH 1;AT 0,0;"E
RROR EN LINEA ": STOP
 260 NEXT z
 270 CLS
 280 PRINT "PON LA CINTA PARA GR
ABAR"
 290 SAVE "COPYCODE"CODE 5E4,144
```

nos permitirá obtener copias de cualquier cosa con una calidad infinitamente superior a la de la rutina anterior.

Copy troceado

El listado 2 es el que nos ayudará a obtener esas «copias perfectas». El truco está en forzar a la impresora a que tenga más resolución que el Spectrum. De esta forma se podrá hacer que una trama corresponda a un solo punto de la pantalla del ordenador, y conseguir gracias a ello la calidad de impresoras más profesionales y mucho más caras. El defecto es que una copia del tamaño que necesitaríamos para obtener la resolución deseada, no cabe en el papel de la impresora, así que la única solución es copiar la pantalla en tres trozos que luego habrá que recortar y pegar. Esta chapuza merece la pena a la vista de los resultados. Con esta rutina se obtienen copias mucho mejores y mucho más grandes y espectaculares. En cualquier caso, el fin justifica los medios.

Funcionamiento de la rutina

La rutina está bastante estructurada en subrutinas sucesivas. Esto facilitó mucho su programación y facilitará muchísimo su comprensión, que de otro modo podría ser complicada, debido a la extraña manera en que está almacenada la pantalla del Spectrum en memoria.

La primera subrutina inicializa el registro HL con la dirección del primer tercio de pantalla y el DE con la del primer tercio de atributos y llama después a la subrutina TRPA, que es la encargada de copiar cada uno de los tres tercios en que se divide la pantalla. Después se deja un espacio en blanco y se calculan las direcciones de pantalla y atributos del siguiente tercio, repitiéndose el bucle si aún no se han copiado todos los tercios y retornando en caso contrario.

La subrutina TRPA copia un tercio completo de la pantalla cuando se la llama teniendo en HL su dirección de pantalla y en DE la de atributos. Para ello realiza un bucle que repite 32 veces para cada uno de los 32 caracteres en

que se divide la pantalla horizontalmente, actualizando cada vez los contenidos de DE y HL para que apunten a la correspondiente columna vertical dentro del actual tercio de pantalla. Una vez copiadas las 32 columnas, retorna. Para realizar la copia de cada una de las columnas se llama a la subrutina UNBY.

La subrutina UNBY copia cada una de las 32 columnas antes dichas. Cuenta con dos bucles anidados. El exterior se repite cuatro veces y el interior 2, por lo que la llamada que se encuentra dentro del bucle interno a la subrutina UNBIT se ejecuta 4 × 2=8 veces, una por cada columna vertical de un *pixel* de ancho que hay en la columna de un carácter de ancho. Por cada columna vertical de un *pixel* en la pantalla hay que producir en la impresora cuatro filas horizontales de un *pixel*, o una fila de cuatro *pixels*, por eso tras haber llamado dos veces a la subrutina UNBIT mediante el bucle interior, habremos obtenido una línea de 8 *pixels*, con lo que habremos llena-

Listado 2 A
CODIGO FUENTE EN ENSAMBLADOR

```
10 ;
20 ;
30 ;  COPY DE GRISES GRANDE
40 ;
50 ;
60 ;  POR PABLO ARIZA-1986
70 ;
80 ;
90        ORG   60000
100 ;
110 ;
120       LD    HL,16384
130       LD    DE,22528
140 LTP   PUSH  DE
150       PUSH  HL
160       CALL  TRPA
170       LD    B,10
180 LOESP PUSH  BC
190       CALL  COBUFF
200       POP   BC
210       DJNZ  LOESP
220       POP   HL
230       POP   DE
240       INC   D
250       LD    A,H
260       ADD   A,8
270       LD    H,A
280       CP    88
290       JR    C,LTP
300       RET
310 ;
320 ;
330 TRPA  PUSH  DE
340       PUSH  HL
350       CALL  UNBY
360       POP   HL
370       POP   DE
380       INC   E
390       INC   L
400       LD    A,L
410       AND   31
420       JR    NZ,TRPA
430       RET
440 ;
450 ;
460 UNBY  LD    B,4
470 LOBI1 PUSH  BC
480       LD    B,2
490       LD    IX,23327
500 LOBIT PUSH  DE
510       PUSH  HL
520       PUSH  BC
530       CALL  UNBIT
540       POP   BC
550       POP   HL
560       POP   DE
570       DJNZ  LOBIT
580       PUSH  DE
590       PUSH  HL
600       CALL  COBUFF
610       POP   HL
620       POP   DE
630       POP   BC
640       DJNZ  LOBI1
650       RET
660 ;
670 ;
680 UNBIT  PUSH  DE
690        PUSH  HL
700        CALL  UNBICA
710        POP   HL
720        POP   DE
730        LD    A,L
740        ADD   A,32
750        LD    L,A
760        LD    E,A
770        JR    NC,UNBIT
780        LD    DE,160
790        ADD   IX,DE
800        RET
810 ;
820 ;
830 UNBICA LD    A,(DE)
840        RRA
850        AND   28
860        PUSH  HL
870        LD    L,A
880        LD    H,0
890        LD    A,(DE)
900        RLA
910        RLA
920        AND   28
930        LD    E,A
940        LD    D,H
950        LD    BC,NEGRO
960        ADD   HL,BC
970        EX    DE,HL
980        ADD   HL,BC
990        PUSH  HL
1000       POP   BC
1010       POP   HL
1020       LD    A,4
1030 LAP   PUSH  AF
1040       LD    A,2
1050 LEAF  EX    AF,AF'
1060       PUSH  BC
1070       PUSH  DE
1080       CALL  MINI
1090       POP   DE
1100       POP   BC
1110       EX    AF,AF'
1120       DEC   A
1130       JR    NZ,LEAF
1140       DEC   IX
1150       POP   AF
1160       DEC   A
1170       JR    NZ,LAP
1180       RET
1190 ;
1200 ;
1210 MINI  RLC   (HL)
1220       INC   H
1230       JR    NC,PAP
1240 RERES LD    A,(BC)
1250       RRA
1260       RR    (IX+0)
1270       RRA
1280       RR    (IX+0)
1290       RRA
1300       RR    (IX+0)
1310       RRA
1320       RR    (IX+0)
1330       INC   BC
1340       LD    A,(BC)
1350          RRA
1360          RR    (IX+32)
1370          RRA
1380          RR    (IX+32)
1390          RRA
1400          RR    (IX+32)
1410          RRA
1420          RR    (IX+32)
1430          INC   BC
1440          LD    A,(BC)
1450          RRA
1460          RR    (IX+64)
1470          RRA
1480          RR    (IX+64)
1490          RRA
1500          RR    (IX+64)
1510          RRA
1520          RR    (IX+64)
1530          INC   BC
1540          LD    A,(BC)
1550          RRA
1560          RR    (IX+96)
1570          RRA
1580          RR    (IX+96)
1590          RRA
1600          RR    (IX+96)
1610          RRA
1620          RR    (IX+96)
1630          RET
1640 ;
1650 ;
1660 PAP      PUSH  DE
1670          POP   BC
1680          JR    RERES
1690 ;
1700 ;
1710 ;
1720 ;         TRAMAS
1730 ;
1740 ;
1750 NEGRO   DEFB 15,15,15,15
1760 ;
1770 ;
1780 AZUL    DEFB 7,15,13,15
1790 ;
1800 ;
1810 ROJO    DEFB 5,15,5,15
1820 ;
1830 ;
1840 MAGENT  DEFB 5,10,5,10
1850 ;
1860 ;
1870 VERDE   DEFB 8,5,2,5
1880 ;
1890 ;
1900 CYAN    DEFB 5,0,5,0
1910 ;
1920 ;
1930 AMARIL  DEFB 4,0,1,0
1940 ;
1950 ;
1960 BLANCO  DEFB 0,0,0,0
1970 ;
1980 ;
1990 ;
2000 COBUFF  EQU  )OECD
```

do todo el BUFFER de impresora, que estará listo para ser enviado a la impresora. Por eso hay que incluir en el bucle externo una llamada a la subrutina de la ROM COBUFF, que se encuentra en la dirección 0ECDh.

La siguiente subrutina es UNBIT. Las subrutinas anteriores necesitaban tan sólo de dos parámetros: dirección de pantalla y dirección de atributo. Esta subrutina y las siguientes necesitarán de un tercer parámetro: la dirección en el *buffer* de impresora, que almacenaremos en el registro indexado IX. Esta dirección será distinta según se esté realizando la primera o la segunda pasada por el bucle interior de la subrutina UNBY. El valor es inicializado en la subrutina UNBY y es modificado al final de la subrutina UNBIT para que sea el correcto la próxima vez que ésta sea llamada. Antes de esto se realiza un bucle de 8 pasadas para copiar en el *buffer* cada uno de los ocho caracteres verticales que componen una columna que tiene como altura la de un tercio de la pantalla. Estos ocho caracteres en vertical habrán de ser convertidos a 32 caracteres en horizontal.

Y llegamos a la subrutina UNBICA. Si ha leído atentamente, podrá deducir que copia una columna de un *pixel* de ancho y de un carácter de alto, que en el *buffer* quedará transformada a una fila de cuatro caracteres de ancho y cuatro pixels de alto. Al principio de esta rutina se toman los colores de papel y de tinta y se buscan las tramas correspondientes, dejando la de tinta en BC y la de papel en DE, registro que podemos utilizar, pues de momento no nos hacen más falta los atributos, aunque tras haber terminado esta subrutina, DE habrá de ser restaurado con la dirección que corresponda. En realidad, casi todas las subrutinas que hemos estado viendo preservan estos registros (DE y HL) en la pila antes de llamar a la subrutina siguiente.

Como vamos a copiar una altura de un carácter, habrá que realizar un bucle de ocho pasadas para cada uno de los ocho *pixels* que tiene un carácter de alto. Pero como cada *pixel* en vertical genera en el *buffer* cuatro *pixels* horizontales, sólo cabrán en un *byte* del *buffer* dos *pixels* de la pantalla (en realidad son cuatro *bytes* lo que ocupan estos dos *pixels*, pues cada uno produce cuatro *pixels* de ancho y cuatro de alto). Así que por cada dos bits que se copian habrá que actualizar el puntero del BUFFER. Por eso en lugar de un bucle de ocho pasadas se realiza un bucle de cuatro pasadas que tiene en su interior uno de dos pasadas. Cada vez que se ejecuta el bucle interior se actualiza el registro IX.

Listado 2 B
CARGADOR BASIC DEL COPY DE GRISES A TAMAÑO GRANDE

```
1 DATA "21004011005BD5E5CD80EA060AC5CDCD0EC110F9E1D1147CC60867FE583BE7C9D5E5CD8FEAE1D11C1585"
2 DATA "2C7DE61F20F2C90604C50602DD211F5BD5E5C5CDAEEAC1E1D110F5D5E5CDCD0EE1D1C110E4C9D5E51780"
3 DATA "CDC2EAE1D17DC6206F5F30F211A000DD19C91A1FE61CE56F26001A1717E61C5F540153EB09EB09E511C1"
4 DATA "C1E13E04F53E0208C5D5CDF2EAD1C1083D20F4DD2BF13D20EBC9CB0624305B0A1FDDCB001E1FDDCB13BC"
5 DATA "001E1FDDCB001E1FDDCB001E030A1FDDCB201E1FDDCB201E1FDDCB201E1FDDCB201E030A1FDDCB400E1D"
6 DATA "1E1FDDCB401E1FDDCB401E1FDDCB401E030A1FDDCB601E1FDDCB601E1FDDCB601EC9D5114C"
7 DATA "C118A40F0F0F0F070F0D0F050F050F050A050A080502050500050004000100000000000254"
200 CLEAR 59999: RESTORE : LET a=10: LET b=11: LET c=12: LET d=13: LET e=14: LET f=15: LET ad=6e4
210 PRINT AT 0,0;"LEYENDO LINEA :": FOR z=1 TO 7: PRINT AT 0,16;z
220 READ a$: LET b$=a$(LEN a$-3 TO ): LET a$=a$( TO LEN a$-4): IF LEN a$/2<>INT (LEN a$/2) THEN PRINT FLASH 1;AT 0,0;"LINEA IMPAR EN": STOP
230 LET w=0: FOR x=1 TO LEN a$ STEP 2: LET v=VAL a$(x)*16+VAL a$(x+1): LET w=w+v
240 POKE ad,v: LET ad=ad+1: NEXT x
250 LET v=0: FOR x=1 TO 4: LET v=v*16+VAL b$(x): NEXT x: IF v<>w THEN PRINT FLASH 1;AT 0,0;"ERROR EN LINEA ": STOP
260 NEXT z
270 CLS
280 PRINT "PON LA CINTA PARA GRABAR"
290 SAVE "COPYCODEgr"CODE 6e4,275
```

Y al fin nos encontramos con la última de todas las subrutinas: MINI. Esta es la que realmente hace el copy de grises en el *buffer*. Las rutinas anteriores sólo servían para calcular en cada momento cuál era el *pixel* de la pantalla que había que copiar y en qué parte del *buffer* había que hacerlo.

Lo primero que tiene que saber es si el *pixel* que va a copiar es de papel o de tinta y seleccionar entonces la trama correspondiente de las dos que ya habíamos buscado. Para ello produce una rotación del contenido de HL, que es el que está señalando al *byte* de la pantalla en el que se encuentra el *pixel*. Esto hace que el *pixel* de más a la izquierda pase al *bit* de acarreo del registro F, lo que nos servirá para realizar un salto condicional y decidir si es necesario cambiar la trama de papel por la de tinta, lo que se hace en PAP. Este método de rotar directamente el contenido de la pantalla hace que la próxima pasada, el *bit* de más a la izquierda sea el que antes era el segundo, y que es por tanto, el que tenemos que copiar. Como esta rotación se realiza ocho veces por cada *byte*, al final el contenido quedará inalterado, pero durante la copia de la pantalla a la impresora se producirá en la primera un efecto curioso, que nos servirá, por otra parte, para saber en cada momento qué parte de la pantalla se está copiando.

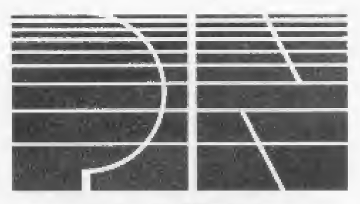
Al punto señalado con el nombre RERES se entra con la dirección de trama en BC y con la dirección de *buffer* en IX. Entonces se va traspasando la trama al *buffer* mediante rotaciones, de tal forma que cuando se introduzca la primera trama que corresponda a una determinada dirección de *buffer*, ésta se introducirá en los *bits* del 7 al 4, y al introducir la siguiente trama, esta primera será traspasada a los *bits* del 3 al 0.

Los resultados de esta rutina pueden ser sorprendentes comparados con los de la anterior. Las dimensiones reales en una GP50S son de unos 23 × 36,5 centímetros, y en la ZX printer algo mayores.

A diferencia del anterior, este programa no es reubicable, así que si no disponemos de ensamblador habrá que utilizarlo en la dirección en la que está colocado, que es la 60000. La forma de emplearlo es igual que la anterior: cargar el código en la dirección 60000, cargar la pantalla y hacer RANDOMIZE USR 60000.

El factor más importante para obtener buenos resultados con ambas rutinas es el de las configuraciones de las distintas tramas.

Pablo Ariza



PIN SOFT, S.A.

Paseo de Gracia, 11 - Esc. C., 2.º 4.ª
Tel. (93) 318 24 53 - 08007 Barcelona

SOFTWARE SPECTRUM

S.I.T.I. V.3*	4.480
Base de datos con cálculos. Al comprar esta versión abonamos 3.000,- ptas. por cualquier versión anterior.	
Context V.9*	4.480
Tratamiento de Textos. Funciona con cualquier impresora. Acentos graves y agudos. **Al comprar esta versión abonamos 3.000 ptas. por cualquier versión anterior.**	
Adaptador SITI-CONTEXT	2.800
Permite pasar información del SITI al CONTEXT.	
M.D.S. - Sistema Operativo para Microdrive	7.000
Conjunto de nuevos comandos BASIC que permiten Acceso Aleatorio a Ficheros en Microdrive con un tiempo medio de acceso de 4 segundos.	
CONTABILIDAD PIN*	3.360
Plan contable 200 cuentas, 2.000 asientos. Hasta 9.000.000.000 Balance con activo-pasivo, cta. resultados. Utiliza el S.O.M.D.S. Cualquier impresora 80 col.	
Kit Utilidades Discovery 1	3.000
10 utilidades CAT extendido. ON ERROR. Compactador de discos, etc.	
AJUSTE DE CABEZALES CASSETTE	2.500
SINTETIZADOR DE VOZ	3.000
MULTI-COPYS (Copys desde 2 cm. hasta 70 cm.)	3.000
COPY GRISES (F+, SP-800, SP-1000, GP-550)	2.800
COPY RS-232	2.800
COPY SERIE RITEMAN F+	2.500
EDITOR 64 (64 columnas en pantalla)	2.750

***Disponible en disco para Discovery 1 al precio de 5.000,- ptas.**

VIDEOJUEGOS

COMANDO	2.443
CRITICAL MASS	1.900
DAM BUSTERS	2.200
FIGHTING WARRIOR	2.100
GYROSCOPE	1.900
HIGHWAY ENCOUNTER	1.900
OLE TORO	2.100
RAMBO	2.100
SABOTEUR	1.900
SGRIZAM	1.950
SUPERMAN	2.760
THE WAY OF EXPLODING FIST	2.300
WEST BANK	1.950

Y TODAS LAS NOVEDADES.
Solicita catálogo.

NOVEDADES

Sistema experto de Flores de jardín	3.500
Sistema experto de Minerales	3.500
APLICACIONES SITI V.3	3.500

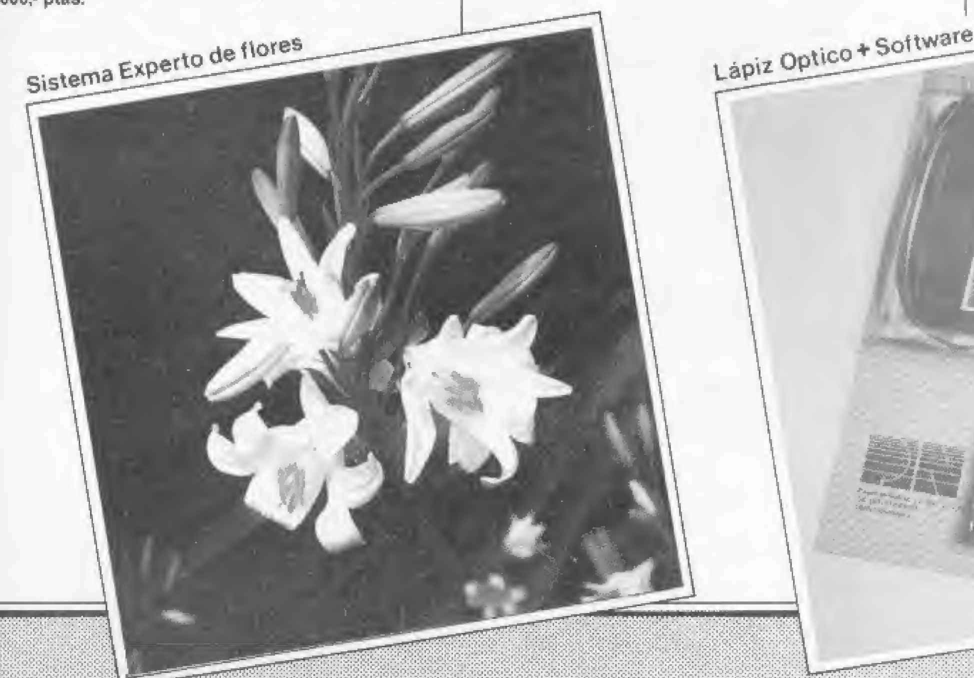
Agenda+Vídeos+Contabilidad doméstica+Stocks, etc. (necesita el SITI V.3)

HARDWARE SPECTRUM

Interface sonido TV	3.920
Interface Joystik	2.000
Joystick Quickshot II	2.912
I/F Centronics	8.000
Lápiz óptico+software	4.850
Interface monitor	3.900
Cinta virgen 15'	112
Monitor Ciaegi f. verde	26.880
Monitor Ciaegi f. ámbar	27.720
Caja para 12 microdrives	112
Teclado Saga 1	10.080
Teclado Saga 3	19.900
Discovery 1+disco Kit	56.000
Diskettes 3 1/2	800
Cable impresora Discovery	3.500
Alimentación ininterrumpida	9.750
Digitalizador de imágenes	39.200
Impresora Riteman F+	80.528

Sistema Experto de flores

Lápiz Optico + Software



TIENDA AL PUBLICO
EN EL CENTRO DE BARCELONA
HORARIO: de 10 h a 20 h. ININTERRUMPIDO
SABADOS CERRADO

IVA INCLUIDO

PEDIDOS POR CORREO O TELEFONO
Envíos contra reembolso a toda España
200 ptas. gastos de envío
En tu domicilio en 3-4 días


COM

GUIA DEL HACKER

MANDO



EL programa seleccionado en esta ocasión para nuestra guía del *Hacker* es el COMANDO. El juego en sí es bastante simple, hay que ir avanzado por una serie de fases en las que nos enfrentamos con innumerables enemigos armados hasta los dientes.

Pero conforme se juega un par de veces, algo nos impulsa a seguir jugando, a averiguar qué es lo que hay en la siguiente fase. Es lo que se llama un juego que produce adicción. Además el número de fases nos desafía con dos cifras. Es una víctima idónea para nuestros POKEs. En principio las ideas de lo que hay que buscar son las de siempre: vidas infinitas, granadas infinitas, eliminar disparos enemigos, eliminar enemigos, inmunidad frente a ellos, pasar directamente de fase, y todo lo que se nos pueda ocurrir durante el análisis del programa.

Como siempre, lo primero es pasarlo a un *microdrive* junto con el monitor. En esta ocasión nos encontramos con un sistema turbo, pero no es muy complicado acceder a él. El programa BASIC se puede parar directamente y contiene escondida detrás de unos códigos de cambio de color la dirección de lanzamiento del cargador, y otra bien visible cuya única misión es despistar en el acceso al código máquina.

Este comienza con una maniobra de distración: una máscara. Tras ella nos encontramos con un cargador turbo convencional, que incluso funciona a la velocidad norr ıal. Se carga todo en un solo bloque que contiene desde la pantalla de presentación hasta el final del programa. La parte importante que debe pasarse a *microdrive* parece encontrarse entre las direcciones 25344 y 65741, y se lanza desde 25630. Con toda esta información sólo queda para poder empezar a trabajar decidir en qué dirección debe cargarse nuestro querido MONS3M2. Tras un par de castastróficos intentos la conclusión es que la 50000 nos viene bastante bien. El programa es demasiado grande como para que pueda coexistir en la memoria junto al monitor sin solaparse. En esta dirección destruye parte de los gráficos del escenario, quedando ilesos los correspondientes a los soldados, con lo que es posible probar algún POKE y lanzar el programa desde el monitor para comprobar el efecto producido en el juego. Los gráficos quedan bastante deteriorados, pero se puede seguir el desarrollo del juego sin muchos problemas. Incluso más tarde descubrí accidentalmente que es posible volver al BASIC desde el programa y después nuevamente al monitor sin tener que cargar nada de nuevo. Esto es debido a que en varios puntos del programa hay rutinas para volver al BASIC devolviendo un código que indica dónde se ha parado. Se supone que la utilidad de estas rutinas era la depuración del programa durante su adaptación.

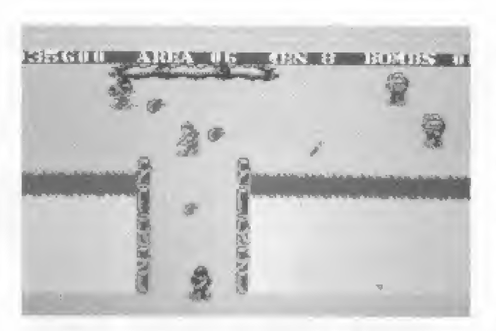
Y casualmente, en un momento inesperado, aparece un *bug* no cazado por los programadores que nos devuelve el control. La forma de utilizarlo es pulsando la tecla K en el mismo momento en que volvemos al menú después de dejarnos matar sin entrar en la tabla de récords (conviene tenerla apretada desde antes). Inmediatamente

se nos pedirá la redefinición de teclas de una forma un tanto extraña y acabará sorprediéndonos con un mensaje o OK... Sin embargo, esto no funciona igual cuando el programa se carga directamente desde la cinta, ya que la forma de salir es colocando la pila como estaba cuando se arrancó el programa desde 641E (25630), se restauran los registros necesarios y se hace un RET, lo que trae consecuencias desastrosas en este caso.

Pero estamos adelantando acontecimientos, y el objetivo es contar las cosas en el orden en que fueron encontradas e intentar explicar qué ideas nuevas pueden utilizarse en cada momento para seguir adelante en la búsqueda.

Empezamos en la dirección 641E y nos encontramos con un salto absoluto a 69D2 seguido de un montón de saltos a otras direcciones. Estas tablas de salto nos recuerdan a otros programas que fueron diseñados genéricamente para cualquier micro que lleve un Z-80 y luego a través de las tablas adaptados a cada máquina concreta. Esta es la primera razón que nos hace pensar que el programa es una adaptación al Spectrum de otra versión.

Pasamos a ver lo que hay en la dirección 69D2. Comienza preparando todo para volver al BASIC, luego inicializa unas variables, llama a unas subrutinas y nos encontramos con una instrucción CP 53 y otra CP 4A. Casualmente, esto coincide con los códigos ASCII de las letras S y J, dos de las tres posibles teclas entre las que se nos da a elegir en el menú principal. Con esto ya tenemos una idea de donde nos encontramos, las subrutinas a las que llama son las encargadas de cada uno de los gráficos de la pantalla de presentación, tabla de récords etc..., y después se queda en un bucle esperando a que se pulse alguna de las teclas posibles, se testean dos y si no es ninguna de ellas es que es la otra.





Anulando las llamadas a las subrutinas se puede averiguar cuál es su función (viendo qué es lo que falta) y así usarlas para diseñar la pantalla para nuestros POKEs. Los descubrimientos más importantes de esta zona son que en la dirección 7615 está la rutina de impresión en pantalla, que escribe lo que hay detrás de la llamada hasta que encuentra el código 1F (esto incluye caracteres de control), y que en la dirección 6BBB es donde realmente empieza a jugarse la partida.

También se descubre en este punto que el programa funciona en modo de interrupción 2, con lo que la rutina en la dirección 7357 se ejecuta periódicamente a razón de 50 veces por segundo. Su función es leer el teclado y

PROGRAMA 1

```
  10 CLEAR 40000: LET n=64256: R
ESTORE
  30 FOR i=1000 TO 1140 STEP 10
  40 READ a$,a: LET s=0
  50 FOR j=1 TO LEN a$-1 STEP 2
  60 LET d=16*(CODE a$(j)-48-7*(
a$(j)>"9"))+CODE a$(j+1)-48-7*(a
$(j+1)>"9")
  70 POKE n,d: LET n=n+1: LET s=
s+d: NEXT j
  80 IF s<>a THEN  PRINT "Error
en la linea ";i: STOP
  90 NEXT i
 100 RANDOMIZE USR 64256
1000 DATA "DD2143FE11BC013EFF37C
D560530F121B751229EFF3E113208FFC
3CDFF2147FB",3884
1010 DATA "11005E018601EDB021005
E22176A2147FB1148FB0100023600EDB
0210A5F22F3",2530
1020 DATA "DE310062C31E6421735F2
2C867CD546721056C3635CD1576160C0
01800564944",2548
1030 DATA "41530D0D494E46494E495
441531FCD4C5F380236B6211BED36C0C
D15764D4554",2669
1040 DATA "52414C4C4554411FCD4C5
F380236402 1F0E23628CD15764D55434
84F530D0D45",2595
1050 DATA "4E454D49474F531FCD4C5
F38023618214AE73625CD15765452494
E4348455241",2518
1060 DATA "530D0D434F4E204449535
041524F531FCD4C5F380236002156643
6C9CD157653",2398
1070 DATA "4F4E49444F1FCD4C5F380
236FE2195DE3628CD1576494E4D4F525
4414C1FCD4C",2918
1080 DATA "5F38023618217D6C3632C
D1576475241 4E414441530D0D494E464
94E49544153",2225
1090 DATA "1FCD4C5F3802363ACD157
656454C4F43494441441FCD5172D6303
8F9FE0A30F5",3127
1100 DATA "3C32857321CE6722C867F
DCB079EC3BB6B3EF7DBFE1F381CFDCB1
56620142A8A",3855
1110 DATA "FD7EFE222328032318F77
E3D2320F2228AFDF1C91F3809F1F1FDC
B07CEC3AE6C",4138.
1120 DATA "1F380DD21004011001B3
EFFCDC604FBC300E1CD517228FBFE4E2
8F7CD5172FE",3821
1130 DATA "532009CD1576205349041
FC9FE4E20ECCD1576204E4F041F37C95
44F444F5350",2789
1140 DATA "45435452554D20504F4B4
5531F",913
```

controlar el sonido del programa. La forma de hacerlo es a través de unas tablas en la dirección 6433. Cuando se quiere emitir algún sonido basta con colocar el dato correcto en la tabla y la rutina se encargará de hacerlo sonar durante el tiempo necesario.

En la rutina de interrupción tiene lugar el primer POKE de nuestra lista: el control de velocidad. En la dirección 7385 está el dato que la controla. El valor normal es 2, y si ponemos un uno el programa se acelera. Cualquier valor mayor ralentiza el juego. Jugando a cámara lenta es más fácil pasar de fase e ir descubriendo otras. De esta forma averigüe que después de las 8 primeras, las fases se van repitiendo cíclicamente, aunque con distintos grados de complejidad. El programa crea adicción de verdad.

Nuestro análisis continúa en la dirección 6BBB. Aquí encontramos nuevas instrucciones CALL y rápidamente, sin entrar en ningún bucle, aparece una llamada a la impresión en pantalla para comunicarnos que hemos pasado de fase, lo cual nos lleva a la conclusión de que la partida se juega en una de las rutinas a las que ha llamado. Las pruebo por orden, aunque desde el principio hay una que se lleva todas la papeletas por estar en una dirección muy distinta de las demás: DC89. Antes de pasar a ese punto veamos qué es lo que se puede encontrar en las otras rutinas. Su principal misión es inicializar las variables del juego. En una de ellas nos encontramos con que coloca en las direcciones FDE6 y FDE7 los datos 5 y 6. Evidentemente tiene que ser el número de vidas y el número de granadas. También aquí se puede intuir que en la dirección FDE5 se guarda el número de fase en que nos encontramos.

El siguiente objetivo es localizar en qué punto se modifica el número de vidas para anularlo, pero para nuestra sorpresa el MONS sólo encuentra el punto en que es incrementado, no cuando es reducido. En seguida surge una idea que nos saca de este atolladero. El registro IY ha sido cargado con la dirección FD80 al principio de todo, y las variables del juego son referidas en muchas ocasiones a este valor. Con esto ya podemos localizar la dirección de nuestro POKE: 6CO5, que nos hace gozar de un número ilimitado de vidas.

En la dirección DC89 está realmente el bucle de juego. En él hay una serie de maniobras y luego una larga serie de llamadas a subrutinas, todas ellas a través de tablas de salto como las localizadas al principio. Cada una de estas llamadas se encarga del movimiento de uno de los posibles objetos en pantalla. La primera (la menos estudiada) debe encargarse de la generación aleatoria de los enemigos y sus

TABLA 1

POKE 25653,182	Vidas infinitas
POKE 26746, 0	No vidas extra
POKE 2666,234	Vida extra cada 1.000 puntos
POKE 26736, 106	
POKE 29573,X	Velocidad
POKE 60699,64	Repetición de disparos
POKE 59217,24	Enemigos inmortales
POKE 62317,24	Enemigos sin granadas
POKE 62337,201	No disparan
POKE 58096,24	Muchos enemigos
POKE 59190,X	Radio de acción del disparo (normalmente 9)
POKE 59252,X	Igual para granadas (normalmente 29)
POKE 59210,0	Permite matar a los de las trincheras a tiros
POKE 59213,0	Igual para el del puente
POKE 59833,201	Sólo se mueve un enemigo
POKE 25686,201	Quita el sonido
POKE 56981,24	Inmortal
POKE 27773,58	Granadas infinitas
POKE 57188,0	Evita la muerte por caer en una trinchera
POKE 58028,24	Evita ser atropellado por el jeep
POKE 58071,201	Igual con la moto
POKE 59319,24	Con los enemigos
POKE 62570,24	Disparos
POKE 62649,134	Granadas
POKE 57896,195	El jeep no dispara pero no muere
POKE 33700,201	No aparece el jepp
POKE 33899,0	No aparece la moto
POKE 34213,0	No aparece el camión

disparos, así como del scroll de la pantalla (es una suposición).

Analizando una de las rutinas descubrí que cuando en la dirección 5BE6 había un dato mayor o igual que 10 significaba que nos habían matado. Rápidamente me puse a buscar en qué punto era introducido ese dato, pero un error en la interpretación de lo que ocurría al principio del bucle de juego me llevo a perder mucho tiempo. El POKE no aparecía. Como este dato en ocasiones era referido al registro IX, trate de buscarlo como (IX + 12). En varias ocasiones parecía que lo había encontrado, cuando cargaba en esta posición un dato mayor que 10, pero siempre conducía a algún tipo de inmortalidad de los soldados enemigos. También trate de buscarlo como punto de colisión con las tablas que controlan los disparos enemigos, pero nada. Durante este tiempo de búsqueda encontré gran cantidad de POKEs mas o menos útiles, como el que permite matar a los soldados atrincherados de un solo disparo. El mejor de todos es el que produce la autorepetición de los disparos, que facilita bastante el juego a los que utilizamos el teclado.

Después de perder bastante el tiempo, descubrí cuál había sido mi error. El dato se modifica al principio del bucle de juego y además es referido directamente. Por fin tenemos la inmortalidad. Es fantástico. Te puede ir a hablar por teléfono mientras te arrojan infinidad de granadas, te disparan, te pisan, que cuando vuelves nuestro héroe sigue en la misma posición y con el mismo número de vidas. El único problema es que somos absolutamente inmortales, las partidas no pueden acabarse nunca. Para solucionar esto se ha incluido en el cargador una rutina que permite abortar el juego pulsando la tecla 2.

El programa 1 es el cargador que contiene los POKEs más interesantes en un menú que aparece justo antes de cada partida. En este cargador también se ha incluido una rutina para pasar directamente de fase en la tecla 1, otra para abortar en la 2, y otra para sacar copias de pantalla sin cabecera en la 3.

Para utilizar los demás POKEs tenéis que introducir el programa 2, lanzarlo y cargar la copia original.

Manuel Arana



Shot®
vendidos del mundo.
QUICKSHOT IV (3 en 1)
Con mando para combate
QUICKSHOT II MSX
Con autodisparo
QUICKSHOT II
Con autodisparo
QUICKSHOT VII MSX
Portátil
Importador exclusivo SVI-España.
SVI
SPECTRAVIDEO





REDADA EN EL RASTRO

Funcionarios del grupo 7.º de la Brigada Regional de la Policía Judicial, con apoyo del grupo 8.º y de la Comisaría de Arganzuela, procedieron durante la mañana del pasado domingo 2 de marzo a la retirada de los programas que se vendían ilegalmente en el Rastro madrileño. El valor de los 11.000 programas retirados supera los 22 millones de pesetas. En el transcurso de la operación, la policía tomó declaración a 28 personas presuntamente implicadas en la apropiación ilícita de los derechos de autor de los programas.

TASWORD THREE

Tasman Software acaba de lanzar el Tasword Three, nueva versión de su conocido procesador de textos. A diferencia del Tasword Two, se suministra en microdrive y no trabaja con cassette. Entre sus características destacan la velocidad y la posibilidad de utilizar 128 caracteres por línea.

```
          TASWORD THREE
       The Word Processor
   © Tasman Software Ltd 1986
           main menu

   Print text file           P
   print with Data merge     D

   Save text file            S
   Load text file            L
   Merge text file           M

   Return to text file       R

   Customise program         C
   save Tasword              T

   catalog/change drive      X

   into Basic                B

0 words      0 chars      Drive 1
1 lines    20977 chars free
```

C I A S



AMSTRAD CPC 6128, uno de los mejores ordenadores según los oyentes de la COPE

Los oyentes de Sábado Chip, el nuevo programa de la COPE y Radio Miramar que se emite los sábados de 5 a 7 de la tarde, han elegido el Amstrad CPC 6128 como uno de los ordenadores más destacados del año. paquetes de *software* profesional de reconocido prestigio se han incorporado al catálogo de programas disponibles para este ordenador: la hoja de cálculo Multiplan, de Microsoft, y la base de datos programable dBase II, de Ashton Tate.

ULTIMA HORA

SERMA, distribuidor oficinal de Konami

Kanji Hiraoka, director de Konani en Europa, se trasladó a Madrid para firmar personalmente el contrato que convierte a Serma en el importador y distribuidor en exclusiva para nuestro país de los productos de la empresa japonesa.

Compatibles Spectrum

Microdigital está produciendo en Brasil un ordenador totalmente compatible con el Spectrum, el TK90X. Sinclair Research se muestra incapaz de detener la producción debido a la caótica situación existente en este país respecto al copyright.

Un portavoz de la compañía manifestó que el ordenador es compatible con el Spectrum y admitió que Microdigital no posee ninguna licencia para fabricarlo.

LENSLOCK: Continúa la polémica

Aunque la protección Lenslock ya ha sido superada por los piratas, sigue implantándose, provocando la protesta de muchos compradores que son incapaces de utilizar los programas protegidos por este procedimiento.

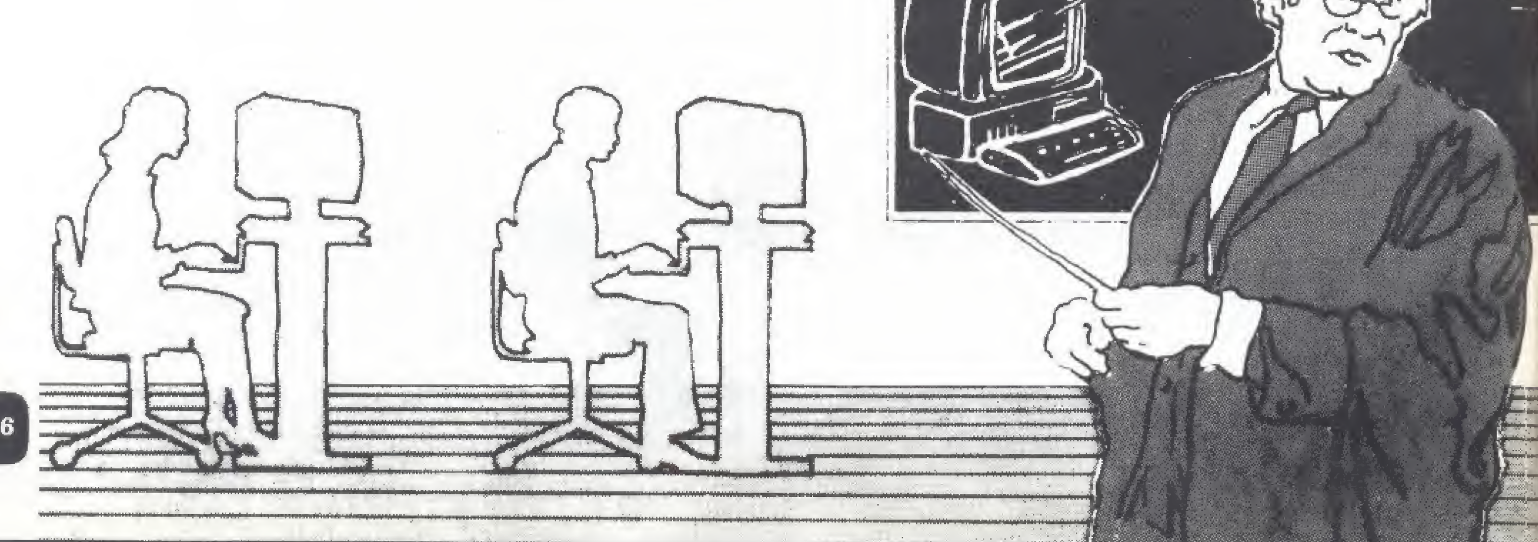
APREN
LENGUAJ

¿Sabría decir cuántos* bits *tiene un* byte*?, ¿podría, ayudado de lápiz y papel, calcular a cuánto equivale en decimal el número binario 00010111?, ¿le suenan a chino las dos preguntas anteriores?

SI la respuesta es SI a las dos primeras preguntas y NO a la tercera está capacitado para seguir leyendo este capítulo, pero si ha conseguido cualquier otra combinación, es el momento idóneo para volver a leer el capítulo primero (algo así como un CALL cap. 1.º). Cuando tenga totalmente claro lo concerniente al sistema de numeración binario podrá adentrarse en el aprendizaje de las instrucciones lógicas y de rotación.

Cuando trabajamos en código máquina, acostumbramos a considerar que el contenido de un registro o una posición de memoria es un valor entre 0 y 255 (o entre −128 y 127). En parte esto es así, pero no debemos olvidar nunca que esos 256 posibles valores que podemos dar a un *byte* corresponden a las 256 posibles combinaciones que podemos conseguir de los ocho dígitos binarios *(bits)* de que está compuesto.

Así como podemos imaginarnos el resultado de la instrucción *assembler* ADD A,25 utilizando el sistema decimal (siempre que tengamos en cuenta que si nos pasamos de 255 comenzaremos de cero y levantaremos la bandera de acarreo), esto nos resultará muy difícil cuando trabajemos con ciertas instrucciones que se efectúan *bit* a *bit*. Por ejemplo, la instrucción OR 240 hace un OR lógico (si nunca ha oído hablar de estas cosas siga avanzando, pronto lo hará) *bit* a *bit* entre el acumulador (que supongamos tiene un 170) y el número 240, quedando el resultado en el acumulador. Si no nos imaginamos operador y operando en forma de dos conjuntos de ocho *bits* difícilmente podremos intuir cuál será el resultado final (ver figura 2).



…DIENDO Capítulo 8
…E MAQUINA

Instrucciones lógicas...

FIGURA 1

		OR
0	0	0
0	1	1
1	0	1
1	1	1

		AND
0	0	0
0	1	0
1	0	0
1	1	1

		XOR
0	0	0
0	1	1
1	0	1
1	1	0

	CPL
0	1
1	0

FIGURA 2

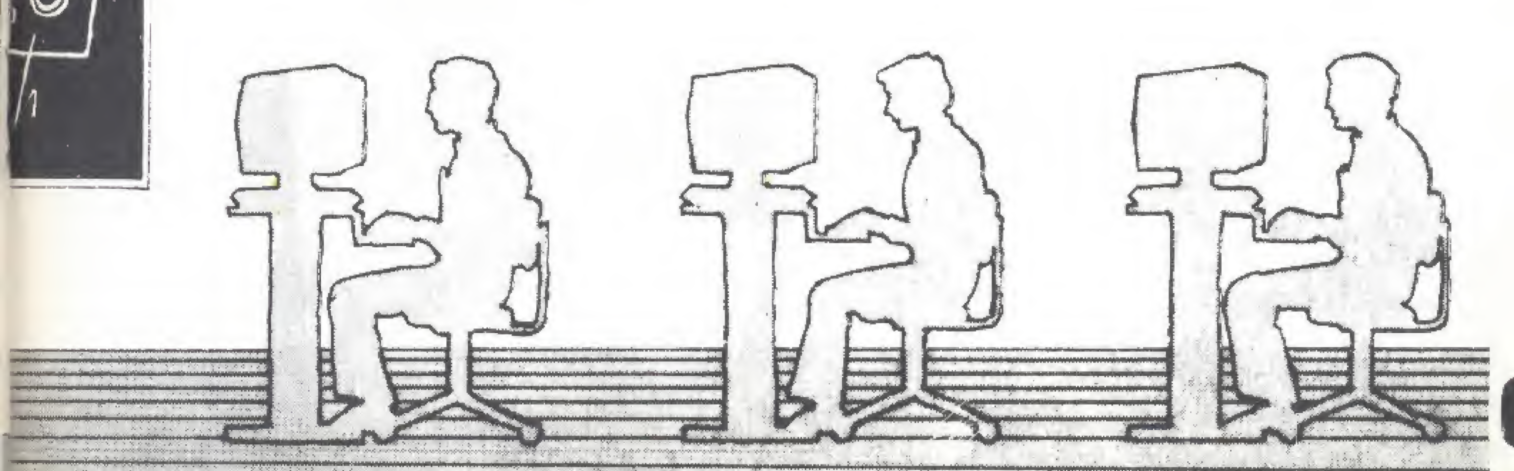
	A	240	OR 240	AND 240	XOR 240	CPL
b7	1	1	1	1	0	[illegible]
b6	0	1	1	0	1	[illegible]
b5	1	1	[illegible]	1	0	0
b4	0	1	[illegible]	0	1	[illegible]
b3	1	0	1	0	1	0
b2	0	0	0	0	0	[illegible]
b1	1	0	1	0	1	[illegible]
b0	0	0	0	0	0	[illegible]

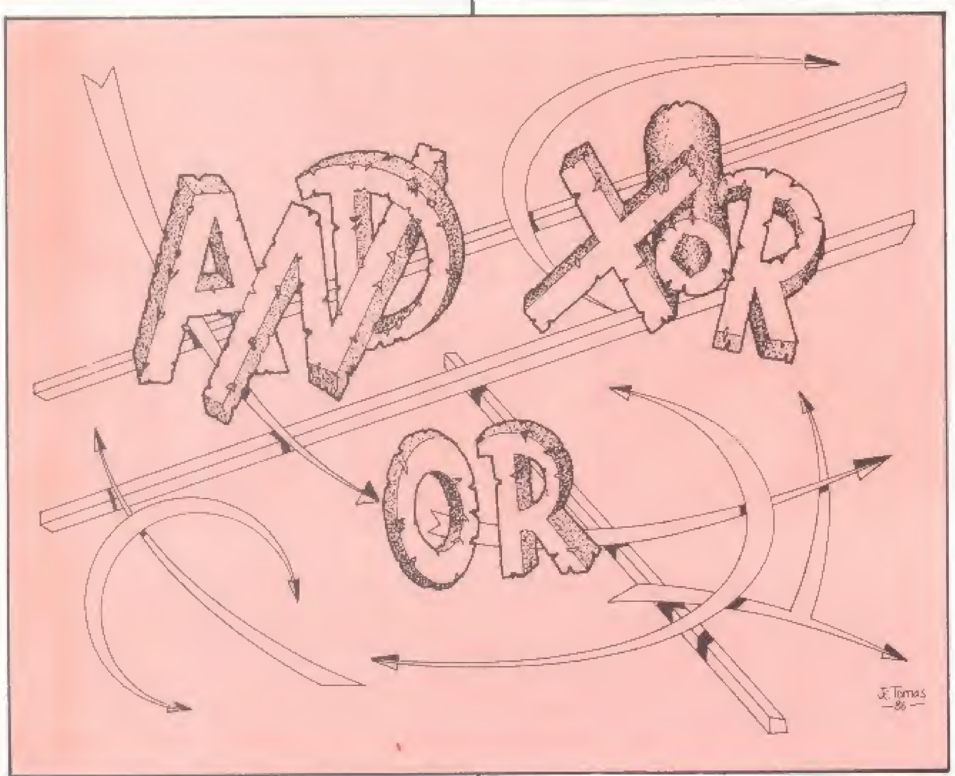
Este tipo de instrucciones, llamadas booleanas en recuerdo de *George Boole* (matemático inglés del siglo pasado que dedicó su vida al estudio del álgebra que lleva su nombre), siempre (excepto con CPL) se llevan a cabo entre el registro A y otro *byte;* este último puede ser: el contenido de cualquier registro (incluido el propio A), un número de ocho *bits* o una posición de memoria direccionada por HL o indexada con IX o IY. Existen cuatro instrucciones que realizan operaciones bien distintas:

OR ("o" o disyuntor inclusivo) compara cada par de *bits* y da como resultado 1 si alguno de los dos valía 1 y 0 si ambos estaban a cero. Esto puede verse más claramente en la tabla de verdad de la figura 1. En castellano equivale a decir: «Esa joven tiene 17 ó 18 años», sólo estaremos mintiendo si la joven en cuestión no tiene ni 17 ni 18 años, pero diremos una verdad como un templo si tiene cualquiera de las dos edades o si tiene las dos (aunque esto último es realmente difícil).

AND ("y" o conjuntor) compara cada par de *bits* y da 1 sólo en el caso de que ambos valgan 1 (ver fig. 1). En castellano sería como decir: «Hoy he comido altramuces y gambas al ajillo», sólo estaremos diciendo la verdad si realmente comimos ambas cosas, pero estaremos mintiendo si sólo hemos comido una de ellas o ninguna.

XOR ("o" o disyuntor exclusivo) compara cada par de *bits* y da como resultado 1 cuando sólo uno de los dos vale 1, y 0 cuando los dos valen 1 o los dos valen 0 (ver fig. 1). En castellano equivaldría a: «¡O me das la cartera o te mato!»,





Las rotaciones simples se diferencian de las circulares en que el bit que sale por un extremo del byte no se copia en el otro extremo, sino que pasa al de acarreo. El contenido previo de éste entra por la derecha o por la izquierda

sólo cumpliremos nuestra palabra si cuando la presunta víctima nos da la cartera nos vamos y le dejamos en paz o si, en el caso de que no nos la diera, nos lo «cargamos» realmente. Seríamos mentirosos además de ladrones si, encima de que nos da la cartera, le pegamos un «navajazo» que le llegue al hígado o si se niega a complacernos y dejamos que se vaya sin hacerle nada.

CPL ("no" o negador) no necesita argumento, actúa siempre sobre el acumulador complementándolo, es decir, poniendo a 0 los *bits* que valieran 1 y a 1 los que valieran 0 (ver figuras 1 y 2); esto equivale a hacer un XOR 255 pero ocupa sólo un *byte.* En nuestro idioma sería como decir: «Eso que dice Ataulfo no es verdad», estaremos diciendo la verdad si lo que Ataulfo dice es mentira, pero estaremos mintiendo si Ataulfo dice la verdad.

COMO SE USAN Y PARA QUE SIRVEN

Después de haber leído pacientemente las líneas anteriores puede haber llegado a la conclusión de que estas instrucciones son muy curiosas pero poco útiles: nada más lejos de la realidad, pues pueden usarse para muchas tareas distintas. La verdad es que es la práctica la que enseña a utilizarlas convenientemente, y cada programador tiene sus «truquis» y se sirve de ellas como más cómodo le resulta; eso sí, hay que reiterar que es imprescindible un perfecto dominio del sistema binario para poder aprovechar su potencia.

La instrucción OR puede usarse para poner a 1 ciertos *bits* del acumulador sin modificar el resto, esto es lo que se suele denominar uso de máscaras. Por ejemplo, si queremos poner a 1 los *bits* 4-7 (nibble superior) del acumulador podemos hacer como en la figura 2 un OR 240 (FOh o 11110000b). De forma similar podemos utilizar AND para poner a 0 los *bits* que deseemos; AND 240 pondrá a cero el *nibble* bajo sin modificar el alto (ver fig. 2). Si lo que queremos es complementar (cambiar el valor) de los *bits* 4-7 habrá que usar XOR 240.

OR también se suele usar para comprobar cuándo en un par de registros hay un cero, esto es necesario porque las instrucciones de incremento y decremento no modifican las banderas cuando actúan sobre registros de 16 *bits.* Para comprobar si HL vale cero podemos hacer LD A, H seguido de OR L, el resultado quedará en la bandera de cero (Z).

Otro uso de estas instrucciones se deriva del hecho de que puedan usar como argumento al propio registro A. XOR A pone a cero al acumulador usando menos memoria que si hiciéramos LD A,O, por el contrario OR A no modifica en nada al acumulador pero sí a las banderas dependiendo del contenido de A, por lo que podremos averiguar si en A vale cero más económicamente que haciendo CP O, además suele utilizarse para desactivar la bandera de acarreo.

OVEJAS DESCARRIADAS

Dentro del juego de instrucciones del Z-80 encontramos algunas instrucciones difíciles de catalogar, como las que actúan sobre la bandera de acarreo. No habrá otro momento mejor que este para echarles un vistazo, ya que en la descripción del trabajo que realizan algunas de ellas se utilizan términos que acabamos de ver, son cuatro:

NEG (complemento a dos), que actúa sobre el acumulador cambiando su signo (en realidad lo que hace es coplementar el *bit* 7). Si trabajamos con números sin signo difícilmente nos resultará útil.

CCF, que complementa la bandera de acarreo, poniéndola a 1 si estaba a 0 y poniéndola a 0 si estaba a 1.

SCF, que pone un uno en la bandera de acarreo.

NOP, que no hace nada, es decir, simplemente deja pasar cuatro estados temporales.

INSTRUCCIONES DE ROTACION Y DESPLAZAMIENTO

Este grupo de instrucciones, que pueden actuar sobre cualquier registro o posición de memoria que direccionemos con HL, IX o IY, permiten hacer rotaciones, rotaciones circulares y desplazamientos aritméticos y lógicos a derecha e izquierda. Pero ¿qué entendemos por «rotar» y «desplazar» un *byte*?

Si consideramos un *byte* como su correspondiente conjunto de ocho *bits*, una rotación circular a la izquierda (RCL) equivaldría a tomar el *bit* 7 y copiarlo en el *bit* de acarreo (bandera de acarreo), después pasar el 6 al 7, el 5 al 6 y así con todos y para finalizar copiar el *bit* de acarreo (lo que había en el 7) en el 0. El resultado de todo esto es como si tomáramos el *bit* 7 y con él «empujaramos» al *byte* por la derecha, copiándolo, para mayor gozo del programador, en la bandera de acarreo.

La mejor forma de llegar a comprender cómo funcionan estas instrucciones quizás sea consultando la tabla de la figura 3, donde la simplicidad de los gráficos hace más intuitivo el aprendizaje.

Las rotaciones simples (RL y RR) se diferencian de las circulares en que el *bit* que sale por un extremo del *byte* no es copiado en el otro extremo, sino que pasa al de acarreo. Es el anterior contenido de este último el que entra por la derecha o por la izquierda.

Los desplazamientos se suelen utilizar para dividir (SRL), dividir números con signo o en complemento a dos (SRA) y multiplicar (SLA) por dos el *byte* usado, quedando en la bandera de acarreo el resto (en las divisiones) o el propio acarreo (en las multiplicaciones).

Aparte de las instrucciones vistas existen dos que deben ser consideradas de distinta forma, pues, más que con *bits*, trabaja con *nibbles*. Son las de rotación decimal, y actúan siempre con el acumulador y una posición de memoria direccionada por HL, rotando el *nibble* bajo del acumulador con los dos de la posición de memoria a derecha (RRD) o izquierda (RLD). El *nibble* alto del acumulador no resulta alterado. Esto es particularmente útil cuando trabajamos en BCD.

FIGURA 3

RLC — Rotación izquierda circular
RRC — Rotación derecha circular
RL — Rotación izquierda
RR — Rotación derecha
SLA — Desplazamiento izquierda aritmético
SRA — Desplazamiento derecha aritmético
SRL — Desplazamiento derecha lógico
RLD
RRD

FIGURA 4

```
 10          ORG   65000
 20
 30          LD    HL,16384
 40          LD    C,192
 50 N_LIN
 60          LD    B,32
 70          OR    A
 80 ROTAR
 90          RR    (HL)
100          INC   HL
110          DJNZ  ROTAR
120          DEC   C
130          JR    NZ,N_LIN
140          RET
```

DESPLAZANDO LA PANTALLA

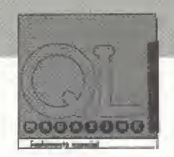
El listado de la figura 4 efectúa un *scroll* lateral de un *pixel* con toda la pantalla. Obsérvese como carga HL con la dirección del primer *byte* del archivo de pantalla, C con el número de líneas, y, ya dentro del bucle N-LIN, B con el número de *bytes* que tiene cada línea (igual al número de caracteres).

Dentro del segundo bucle (ROTAR) se hace una rotación derecha del *byte* direccionado por HL, con lo cual el *bit* que sale por la derecha queda en la bandera de acarreo. Después de incrementar HL se cierra el bucle, con lo que se vuelve a ejecutar la rotación con el resultado de que el *bit* que había quedado en la bandera de acarreo entra por la izquierda del nuevo *byte* rotado. Ahora es cuando podemos darnos cuenta de por qué se hace un OR A para cada línea, esto pone a cero el *bit* de acarreo para que no entre nada por la izquierda de la pantalla (si suprimimos esta instrucción conseguiremos «rotar» la pantalla, es decir, lo que salga por la derecha entrará por la izquierda).

Fácilmente podríamos modificar esta rutina para que haga un *scroll* a la izquierda. Para ello bastará con sustituir 16384 por 22527 (línea 40), que es el último *byte* del archivo de pantalla, decrementar HL en lugar de incrementarlo (línea 100) y cambiar RR por RL (línea 90) para que la rotación se haga hacia la izquierda.



Compilación de SuperBASIC



68008 VERSUS Z-80

Existen muchos usuarios con cierta experiencia en código máquina del microprocesador Z-80, debido sin lugar a dudas al gran auge del Spectrum. Algunos de estos usuarios, al adquirir un QL, se encontraron con un mundo completamente nuevo, ya que el Motorola 68008 poco tiene que vez con el familiar Z-80. Programar directamente el 68008 no consiste sólo en tener acceso a unos bytes de más y a una potencia operativa mayor, sino una filosofía de programación completamente nueva.

Los primeros microprocesadores que se construyeron estaban enfocados a tareas muy determinadas, con un juego de instrucciones reducido. El **Intel** 8080 puede considerarse el microprocesador capaz de resolver una gama mayor de problemas con el mismo juego de instrucciones.

El 8080 era un microprocesador de 8 *bits*, y aunque la compañía le retocó para que pudiera trabajar con canales de 16 *bits*, no había suficiente espacio en memoria para operar con ellos. Debido a estas limitaciones creó el Z-80, al cual se le añadieron 450 instrucciones oficiales, aunque durante el proceso de diseño, surgieron alrededor de una centena de «no oficiales» que no se mencionan en ningún manual, como puede ser el conjunto de instrucciones SLL *(Shift Left Logical)* que brillan por su ausencia en la página 184 del manual del Spectrum (30H a 37H).

El Z-80, aunque sigue siendo un microprocesador de 8 *bits*, puede operar con 16 *bits* en el *bus* de direcciones. (De esta forma es capaz de direccionar 2^{16} (65536) celdas de memoria).

68008 CONTRA Z-80

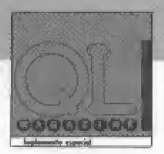
El **Motorola** 68008 es una versión de 8 *bits* de su antecesor el 68000 de 32 *bits*. Mientras el Z-80 fue diseñado para abarcar una gran cantidad de operaciones (es uno de lo más completo del mercado), el 68008 (así como el 68000 y toda su familia) está orientado hacia la simplicidad de la programación.

En el Z-80 cada uno de los registros tiene su especial idiosincracia. Por ejemplo, para los *ports* de entrada-salida sólo puede usarse el registro C; en bucles rápidos, el B; IX e IY son los únicos que aceptan índices; para operaciones aritméticas el A, y así sucesivamente. Sin embargo, tanto el 68008 como todos los miembros de su familia pueden operar indistintamente con cualquier juego de registros para lo que deseemos hacer.

DISTINTOS REGISTROS

El Z-80 trabaja indistintamente con registros de 8 y 16 *bits*, poseyendo estos últimos una conformación particular (están constituidos por 2 registros de 8 *bits*, siendo el 1.º el menos significativo el 2.º el más significativo). Los registros de uso general (de 8 *bits*) son: B, C, D, E, H y L además del acumulador (registro A), siendo los dobles BC, DE, HL y AF. Aparte de todos éstos, existe un grupo de Registros Alternativos y los registros índices IX e IY así como el I (dirección de página de interrupción) y R (refresco de memoria).

En cuanto al 68008, tiene 8 registros de direcciones (A0 a A7) que contienen las direcciones de memoria que está utilizando el programa, y otros 8 registros de datos (D0 a D7) que calcula y almacenan datos en memoria. Aparte

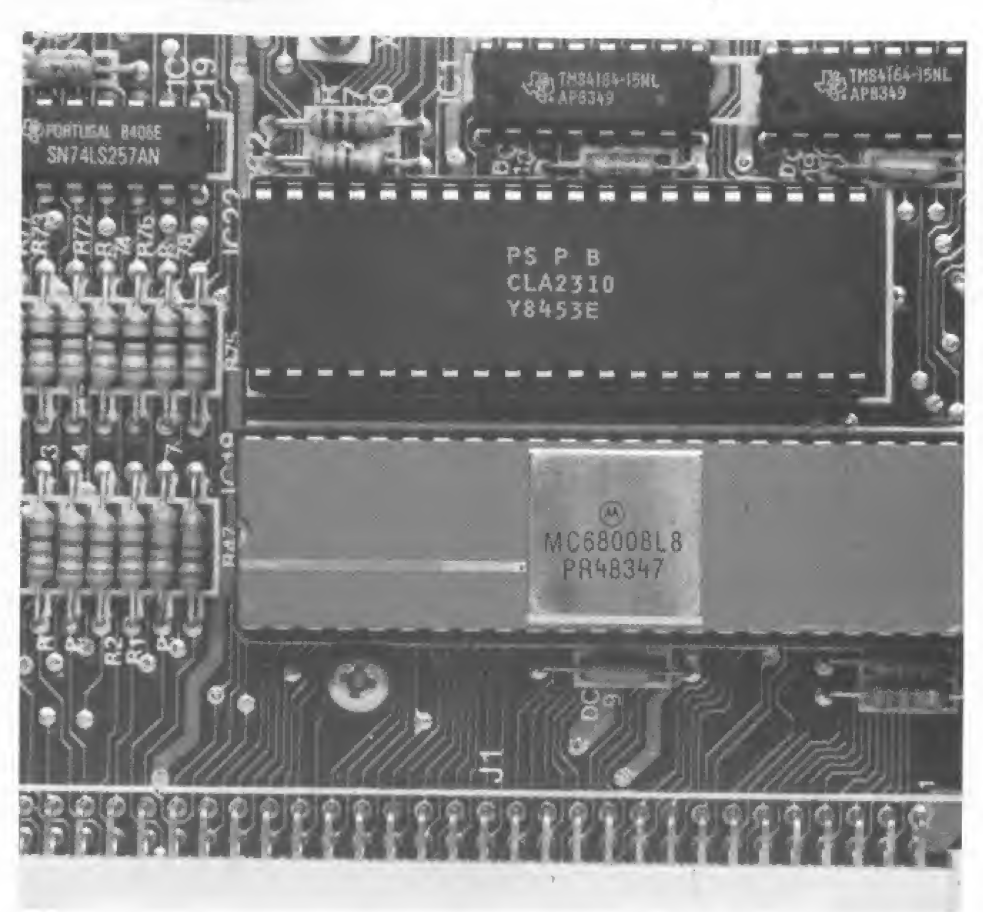


de estos dos juegos de registros, también tiene un contador de programa y un registro de estado. Todos los registros utilizados tienen una longitud de 32 *bits*, menos el registro de estado con sólo 16. Esto hace que la programación en código máquina de 68008 sea muy consistente y facilita el aprendizaje del juego de instrucciones. Aquí ya no es necesario enfrentarse con las sutilezas del Z-80, y la simetría resultante hace más fácil la depuración de errores, alteración de programas y creación de compiladores. En este último sentido es mucho más difícil escribir un buen compilador para el Z-80 debido a la amplia gama de casos especiales que deben ser reconocidos en vistas de producir el mejor código objeto posible. En el extremo, un buen compilador para el 60008 puede producir un código ha partir de un lenguaje de alto nivel como el «C» casi tan eficiente como uno que se hubiese escrito a mano. Esta es la razón por la que el QDOS y el sistema operativo UNIX se han podido escribir enteramente con un compilador en lugar de un ensamblador. (Los programas compilados se escriben más fácilmente que los ensamblados).

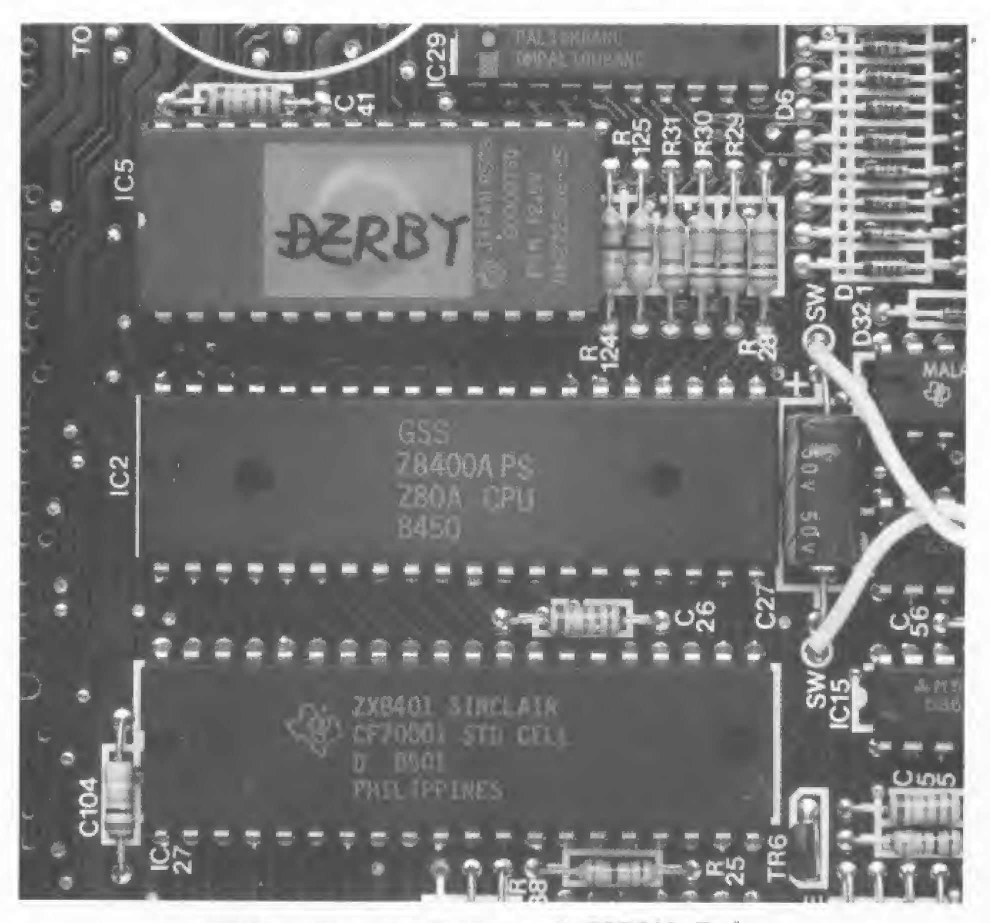
Instrucciones complejas

En teoría, un micro de 16 *bits* es dos veces más potente que uno de 8 *bits*, ya que puede procesar dos veces más información en un mismo paso. Sin embargo, en la práctica, la diferencia es mucho más grande debido a la necesidad de emplear pasos intermedios para convertir dos datos de 8 *bits* en un resultado de 16 *bits*. A diferencia del Z-80, el 68008 tiene la facultad de producir un resultado de 16 de forma inmediata a pesar de que sólo transfiere 8 *bits* a un mismo tiempo cada vez que transmite información.

El 68008 tiene incorporado en su juego de instrucciones algunas muy complejas que hacen en un sólo paso ciertas tareas en las cua-



El 68008 se reconoce fácilmente en el QL por la chapa de oro que lo cubre.



Z-80 en el Spectrum 128, junto a la EPROM «Derby».



■ **El juego de instrucciones del 68008 está orientado hacia la facilidad de programación.**
■ **El 68008 es capaz de hacer una división o una multiplicación de 32 bits en un solo paso frente al centenar que requiere el Z-80.**
■ **La simetría del 68008 hace más fácil la depuración de errores, alteración de programas y creación de compiladores.**

LA CPU 68008

BYTE ALTO
REGISTRO DE PROPOSITO GENERAL E INDICE
BYTE BAJO

BYTE ALTO
REGISTRO DE PROPOSITO GENERAL
BYTE BAJO

BYTE ALTO
PUNTERO DEL STACK
BYTE BAJO

BYTE ALTO
CONTADOR DE PROGRAMA
BYTE BAJO

BUS DE DIRECCIONES
BUS DE DATOS
UNIDAD ARITMETICO-LOGICA
BUS DE CONTROL
REGISTRO DE ESTADO DEL PROCESADOR
REGISTRO DE DIRECCIONES
REGISTRO DE DATOS

68008

El 68008 es el último de una serie de procesadores compatibles, con similares juegos de instrucciones y registros. Un conjunto de registros índices guarda las direcciones de memoria que está utilizando el programa y un conjunto de registros de datos se utiliza para calcular y almacenar datos. Todos los registros son de 32 bits, excepto el registro de estado, de 16. Los cálculos se realizan en la unidad de aritmético-lógica (ALU), que almacena 32 bits de instrucciones de una vez. El puntero del stack y el contador de programa mantienen las dirección de las áreas de memoria utilizadas por el ALU para las instrucciones de programa y los datos. Aunque los buses de datos y direcciones dentro del microprocesador son de 32 bits, el 68008 transfiere 8 bits de datos y 20 de direcciones a la vez al mundo exterior. El 8 al final del número que designa al procesador se debe precisamente al número de bits de datos que puede transferir de una vez.



LA CPU Z-80

BUS DE DIRECCIONES
BYTE ALTO — REGISTRO INDICE — BYTE BAJO
BYTE ALTO — REGISTRO INDICE — BYTE BAJO
BYTE ALTO — PUNTERO DEL STACK — BYTE BAJO
BYTE ALTO — CONTADOR DE PROGRAMA — BYTE BAJO
REGISTRO DE DIRECCIONES
BYTE ALTO — REGISTRO DE PROPOSITO GENERAL — BYTE BAJO
BYTE ALTO — REGISTRO DE PROPOSITO GENERAL — BYTE BAJO
BYTE ALTO — REGISTRO DE PROPOSITO GENERAL — BYTE BAJO
REGISTRO DE DATOS
BUS DE DATOS
ACUMULADOR
UNIDAD ARITMETICO-LOGICA
BUS DE CONTROL
REGISTRO DE ESTADO DEL PROCESADOR

Z-80
El Z-80 es una versión mejorada del 8080 que incluye un juego de instrucciones más versátil y un conjunto «de reserva» de sus principales registros. Todos los registros son de 16 bits, excepto el acumulador, de 8 bits. (Los otros 8 bits se utilizan para el registro de estado o de banderas).
IX e IY se utilizan como registros índice de tablas. El puntero stack y los registros de programa mantienen las direcciones empleadas por la ALU para el programa y el almacenamiento de datos. El Z-80 es único, pues tiene también un registro de 8 bits de interrupciones y un registro de refresco. El registro de interrupciones contiene los 8 bits superiores de una dirección de interrupción, mientras los ocho inferiores los proporcionan los periféricos. El registro de refresco se encarga del refresco de memoria, eliminando así otro chip externo.

les un microprocesador más sencillo invertiría muchos más. Por ejemplo, el 68008 puede realizar multiplicaciones y divisiones de 32 *bits* en un único y rapidísimo paso, cuando el Z-80 necesitaría emplear alrededor de un centenar para hacer la misma operación, al no tener implementadas estas funciones.

El 68008 tampoco sacrifica flexibilidad al utilizar sus registros de 32 *bits*, pues cada instrucción puede emplearse en la forma de 32, 16 y 8 *bits*.

ACCESO AL C/M DESDE EL BASIC Y EL SUPERBASIC

En el Spectrum, la forma más fácil de ejecutar un programa en código máquina consiste en el uso de la instrucción USR «dirección de comienzo», habiendo antes protegido la zona de memoria donde va almacenado el código mediante CLEAR. Sin embargo, el QL no utiliza la llamada USR, sino CALL seguido de un conjunto de parámetros opcionales. Mediante la instrucción RESPER (X) se reservan X *bytes* de memoria.

Un programa de QL que utilice código máquina puede tener la siguiente apariencia general:

```
1000 REM RESERVA AREA
1010 LET A = RESPR (100)
1020 PRINT "RESERVADOS 100 BYTES DESDE»; A
1030 FOR X = 1 TO 100
1040 READ D: POKE A + X, D
1050 NEXT X
1060 CALL A, D0, D1, D2, D3, D4, D5, D6, D7, A0, A1, A2, A3, A4, A5
1070 REM VUELVE SI D0 = 0
1080 DATA 1, 2, etc.
```

Finalmente, otra gran diferencia entre el 68008 y el Z-80 radica en el modo de leer el primero los datos almacenados en memoria. El 68008 emplea la técnica llamanda *pipelining*. El Z-80 (y cualquier otro microprocesador que no lleve incorporada dicha técnica) trabaja realizando las siguientes tres acciones una por una y en el orden descrito: búsqueda de instrucción, decodificación y ejecución de la instrucción. Sin embargo, el 68008 usa una unidad de separada para leer instrucciones de tal manera que pueda seguir leyendo las siguientes instrucciones mientras el resto del microprocesador está ejecutando la última. Este sistema permite aumentar de manera eficiente la velocidad de proceso.

Orlando Araujo Martín



Compilación de SuperBASIC

Antes del lanzamiento del QL a la calle, se decía que el SuperBASIC incorporado en él, iba a ser un lenguaje potente, multitarea y que trabajaría a una velocidad constante sin tener en cuenta el tamaño del programa. Después vino la gran desilusión: el SuperBASIC no era multitarea, resultaba más bien lento y sólo tenía 7 dígitos de precisión. Para remediar esto, en julio de 1984 Digital Precision empezó a trabajar en el SuperCharge, un compilador automático destinado a convertir el SuperBASIC en rapidísimo código máquina...

Digital Precision pretendía resolver todos los problemas, sin afectar para nada la potencia del lenguaje.

Supercharge debería compilar la inmensa mayoría de los programas existentes en SuperBASIC, sin ninguna alteración. También tendría que ser capaz de funcionar en todas las versiones de QL existentes, con un consumo mínimo de memoria, y el código objeto generado debería ejecutarse a la máxima velocidad posible. De esta manera, se llevó a buen término el diseño del **SuperCharge.**

El SuperBASIC es lento porque tiene que buscar cada comando, línea y variable de un programa donde quiera que estén. Es un lenguaje interpretado y por lo tanto cada punto y coma, cada instrucción, son verificados detenidamente mientras se ejecuta el programa.

EL ANALIZADOR DEL LENGUAJE

El compilador **SuperCharge** está formado por dos partes principales: el *parser*, que analiza las instrucciones del programa original en SuperBASIC, y un generador de código objeto, que convierte las instruccioones a las correspondientes en código máquina. Le acompañan también un programa de demostración y un conjunto de procedimientos para el control de tareas y depuración de errores.

El *praser* fue inicialmente escrito en 2.500 líneas de SuperBASIC sin un sólo GO TO o GOSUB. ¡Su primer gran desafío fue compilarse a sí mismo!

El *parser* lee el programa que va a ser compilado y el resultado de sus análisis se almacena en una zona libre de la RAM del QL. Esto produce una rápida compilación y deja libres 40 K de memoria (en el QL *estándar*) para que el programa que se va a compilar pueda caber perfectamente.

EL GENERADOR DE CODIGO

El generador de código es un programa de poca longitud que se carga sobre el *parser*. Está escrito en código máquina para que pueda tratar *bits* y *bytes* a la máxima velocidad.

Este generador código selecciona las rutinas en código máquina que realizarán la compilación. El código producido se graba en un *diskette* o *microdrive*, de tal modo que pueda ser cargado con EXEC. Los programas compilados se car-



El compilador de Digital Precisión está protegido con Leuslock.

gan rápidamente, ya que no tienen que ser «tokenizados» por el SuperBASIC. Debido a estos programas no pueden listarse, están a salvo de ojos indiscretos y de los intentos de piratería.

VENTAJAS Y LIMITACIONES DE SUPERCHARGE

Existen algunas diferencias entre programas compilados e interpretados además de la velocidad.

En el *SuperCharge*, la primera diferencia que se observa es la representación numérica con 9 dígitos de precisión.

SuperCharge se adopta automáticamente a diferentes ROMs y juegos de comandos (transcurren unos momentos antes de la ejecución mientras busca las rutinas necesarias).

El compilador solamente prohíbe unas pocas instrucciones (como ED y RENUM) que cuentan con la presencia de las estructuras de datos del intérprete. Si tuviese que estar al corriente de la fuente del programa y la lista del nombre, sería apenas un poco más veloz que el intérprete.

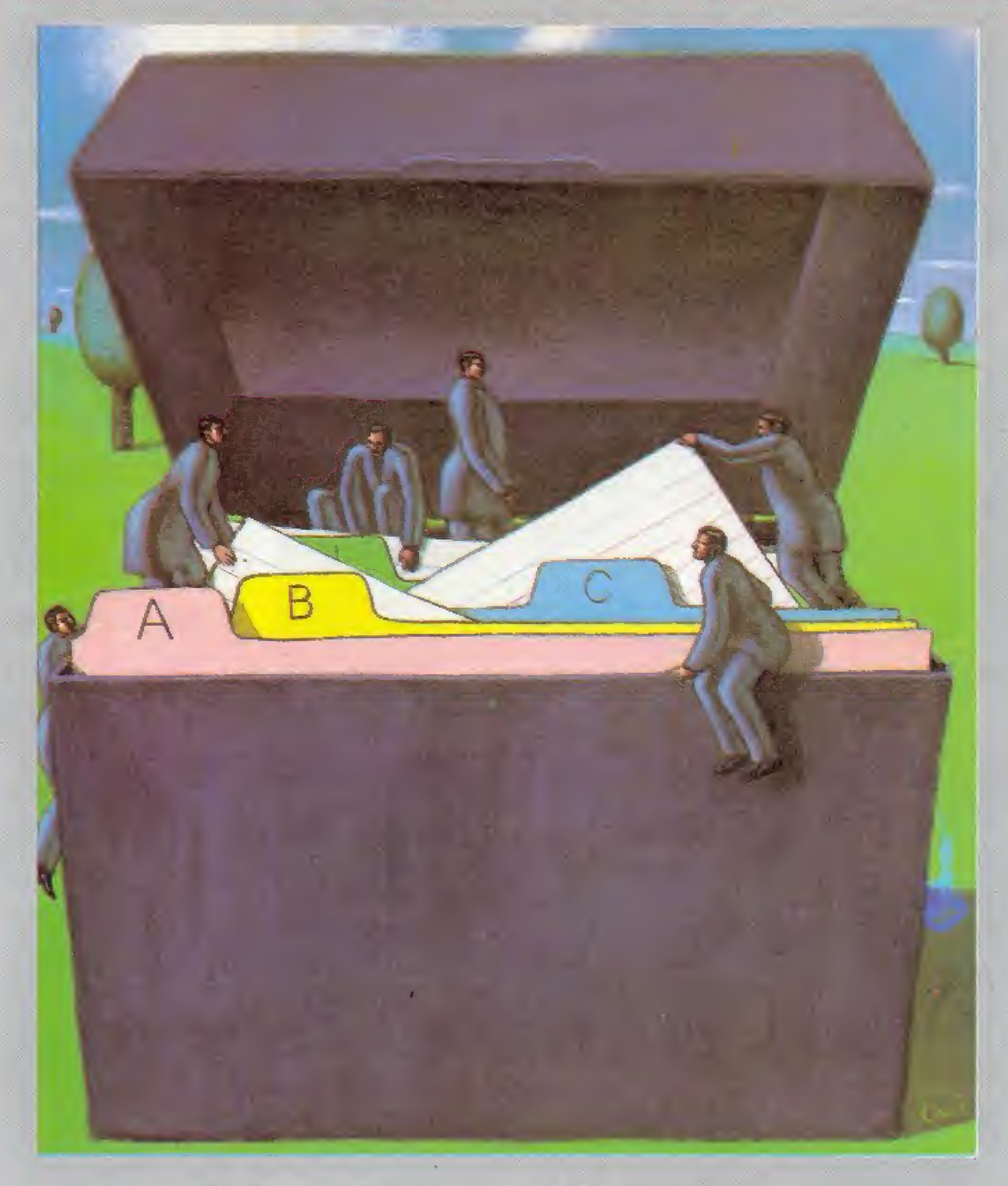
Otra limitación consiste en que una matriz no debe de contener más de 32.767 elementos. Cadenas y números enteros siempre deben estar señalizados con un signo *dollar* o un tanto por ciento. De la misma manera, los parámetros de GO TO, GO SUB y DATA tienen que ser constantes (no expresiones).

Estas escasas limitaciones son necesarias para que pueda resultar más compacto y rápido. Si se comete algún error, el compilador muestra un mensaje en inglés que la informa y le muestra el lugar exacto del programa donde se produjo el error.

Los programas con **SuperCharge** funcionan a una velocidad nueve veces más rápida que los programas SuperBASIC convencionales. Esta velocidad puede incluso ser cuadruplicada si se excluye en el programa la coma flotante.

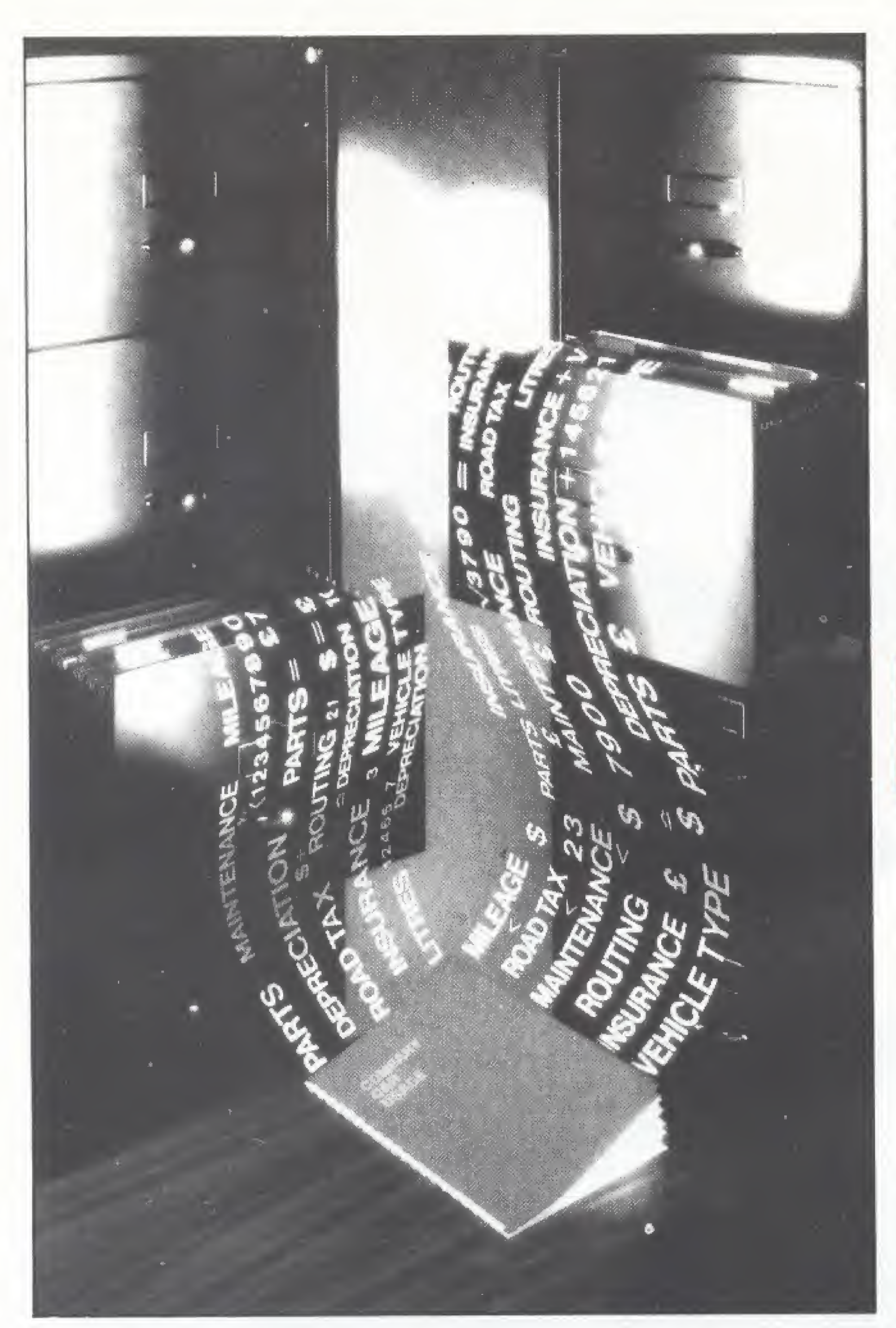
Orlando Araujo Martín

FICHEROS SECUENCIALES EN
PASCAL



Uno de los principales defectos del Pascal es su incapacidad para trabajar con ficheros. Las rutinas desarrolladas por Santi Casablancas resuelven este inconveniente, permitiendo la utilización de ficheros secuenciales en microdrive desde el compilador Pascal Hisoft.

Los procedimientos y funciones utilizadas se describen por orden de importancia, excepto ERROR, que se declara FORWARD al principio y se codifica al final.



Lo primero que debe hacerse es declarar una serie de tipos y constantes utlizados en estas rutinas para evitar errores fatales para el sistema.

Las constantes son:
DSTR1 = 23766 número Microdrive
DSTR11 = 237767 segundo byte núm. Microdrive
SSTR1 = 23768 número canal
LSTR1 = 23768 identificador de canal
NSTR1 = 23770 longitud nombre
TSTR1 = 23772 dirección nombre
CHANS = 23631
STRMSR = 5c16H inicio área corrientes
CURCHL = 23633 dirección de destino de I/O

Los tipos son:
FILENAME = ARRAY (1.. 10) OF CHAR;
DRIVENUM = 1... 8;
CHANEL = 1... 15;
FILE = tipo de la variable a utilizar en el archivo.

Procedimiento OPEN: (NOMBRE, CANAL, DRIVE):

Es el procedimiento utilizado para abrir un archivo en Microdrive y asignarle una corriente. Los parámetros son NOMBRE: FILENAME, CANAL: CHANE (normalmente del 4 al 15) y DRIVE: DRIVENUM.

Primero se calcula la dirección de la corriente y se comprueba que no esté abierta (se provoca un error si ya está asignada a un canal). Seguidamente se pokean la dirección del nombre de fichero, su longitud y el número del Microdrive en las variables del sistema D_STR1, N_STR1 y T_STR1 y se ejecuta una rutina en lenguaje máquina (contenida en un procedimiento INLINE) que crea un canal M y devuelve en la variable local AUX el desplazamiento de canal. Finalmente se pokea el desplazamiento en la dirección de corriente calculada al principio.

Tiene los mismos efectos que el OPEN del BASIC. Si el archivo no existe, lo crea y si existe lo abre para lectura.

Procedimiento CLOSE (CANAL):

Utilizado para cerrar un archivo. Es obligatorio sobre todo en los archivos de escritura.

El único parámetro utilizado es CANAL: CHANEL. Puede dar dos errores; canal no abierto o que el canal no sea de Microdrive.

El procedimiento OPEN tiene los mismos efectos que la instrucción BASIC equivalente. Si el archivo no existe lo crea, si existe lo abre para lectura

Primero se calcula la dirección del canal y se comprueba las condiciones de error para después ejecutar una rutina en código máquina (contenida en un procedimiento INLINE) que cierra el canal. Finalmente, se pone a cero la dirección de la corriente.

Los efectos son iguales que la sentencia CLOSE del BASIC.

Procedimiento COUT (CANAL, CHAR):

Utilizado para escribir un carácter en el canal especificado.

Los parámetros son CANAL: CHANEL y CAR: CHAR. Puede comprobar que contiene otro procedimiento en ámbito local, el procedimiento SPOUT. Si ya lo tiene definido anteriormente, puede eliminarlo.

Primero se calcula la dirección del canal y se pokea en la variable del sistema CHUCHL. Seguidamente se emplea el procedimiento SPOUT para escribir el caracter en la corriente abierta, para finalmente, devolver a CHURCHL su valor primitivo.

No se comprueba si la corriente está abierta o si es de Microdrive para permitir su empleo con cualquier canal.

Función CIN (CANAL):

Esta función devuelve el siguiente caracter existente en la corriente especificada.
El parámetro de entrada es CANAL: CHANS y la salida entrega una carácter. Es parecida al INKEY$ del BASIC.
Consiste solamente en un segmento de código máquina contenido en un procedimiento INLINE.

Función LECOESC (CANAL):

Devuelve un valor TRUE si el archivo es de escritura y FALSE si es de lectura..
El parámetro de entrada es CANAL: CHANEL y devuelve un valor BOOLEAN.
Primeramente se calcula la dirección del canal para sondear el flag de escritura (bit 1 del byte 24 del canal). Se utiliza el procedimiento ODD para aislar el bit 1.
Si la corriente no está abierta o el canal no es de Microdrive, se genera un error.

Función EOF (CANAL):

Devuelve un valor TRUE si se han leído todos los caracteres del archivo especificado y FALSE en otro caso.
El parámetro de entrada es CANAL:CHANEL y devuelve un valor de tipo BOOLEAN.
Se empieza por calcular la dirección del canal y verificar las condiciones de error (no abierto o que no sea de Microdrive). Después se carga la variable local A con el *byte* 67 del canal y se ejecuta una rutina en código máquina para enmascarar todos los *bits* menos el 1. Finalmente, se comprueba el estado de este *bit* y si el contador de *bytes* leídos es igual al de *bytes* existente. Si se dan estas condiciones es que la próxima lectura provocará un System Call error (eof).

Procedimiento ERASE (NOMBRE, DRIVE):

Este procedimiento borra el archivo NOMBRE del drive DRIVE. Tiene los mismos efectos que la sentencia ERASE del BASIC.
Los parámetros de entrada son NOMBRE: FILENAME y DRIVE: DRIVENUM.
Se empieza por pokear la dirección del nombre, su longitud y el número de drive en las variables del sistemas D_STR1, L_STR1 y N_STR1. Seguidamente se ejecuta una rutina en código máquina (contenida en un procedimiento INLINE) borra el archivo (una sola vez, aunque esté grabado varias veces).

Procedimiento RESET (NOMBRE, CANAL, DRIVE):

Es el primero de los procedimientos complejos. Básicamente comprueba que la corriente especificada no sea utilizada por otro dispositivo. Si no lo está, la cierra si estaba abierta y la reabre para lectura. Si el archivo no existe en Microdrive se genera error de fichero no encontrado.
Tiene utilidad para retroceder al principio de los ficheros de lectura, para acabar una escritura y pasar a lectura, etc.
Los parámetros utilizados son NOMBRE: FILENAME, CANAL: CHANEL y DRIVE: DRIVENUM.

Las variables locales son direccionales mediante el registro IX, asignado al principio de cada bloque interno

Procedimiento REWRITE (NOMBRE, CANAL, DRIVE):

Se utiliza para crear o reescribir un fichero. Comprueba que el canal no sea utilizado por otro dispositivo y lo cierra en caso de que esté abierto. Seguidamente abre el canal y comprueba que sea de escritura. En caso de que sea de lectura (fichero existente en el Microdrive) lo cierra y lo borra. Repite el proceso hasta que consigue un canal de escritura. (Por tanto, borra copias múltiples).

Procedimiento WRITEF (CANAL, FVAR):

Utilizado para escribir un dato del tipo FILE en la corriente CANAL.
Los parámetros son CANAL: CANEL y FVAR: FILE.
Se comprueba que el canal esté abierto, sea de Microdrive y sea de escritura. Después se traspasa caracter a caracter el dato al Microdrive mediante el procedimiento COUT. Finalmente se escribe un CHR (13) para hacerlo compatible con el BASIC.

Procedimiento READF (CANAL, FVAR):

Su utiliza para llenar la variable FVAR: FILE con el próximo dato del Microdrive especificado.
Se comprueba que el canal esté abierto, sea de Midrodrive y de lectura y después se llena la variable mediante pokes y funciones CIN.
Si se intenta leer más allá del fin de fichero se generará un system call error (eof).
El parámetro CANAL: CHANEL es del tipo valor, mientras que el FVAR: FILE es del tipo variable.

Procedimiento APPEND (NOMBRE, CANAL, DRIVE):

Permite añadir información al final de un fichero existente.
Los parámetros son NOMBRE: FILENAME, CANAL: CHANEL Y DRIVE: DRIVENUM.
Primero se comprueba si el canal está abierto y en caso de que no lo esté, se abre. Si el canal pertenece a otro dispositivo, se genera un error.
Si el fichero abierto es de escritura, la cosa acaba aquí, pero si es de lectura, se ejecuta una rutina en código máquina (contenida en un procedimiento INLINE) que busca el último sector del archivo y lo prepara para añadir datos. El canal pasa a ser de escritura.

Procedimiento FORMAT (NOMBRE, DRIVE):

Es el primero de los auxiliares. Es idéntico al FORMAT «M» del BASIC. No se puede utilizar si se tiene abierto un canal de Microdrive.
Se utiliza un curioso sistema de ubicar el código máquina en un array de caracteres, ya que es dependiente de la posición (no relocalizable) y se calcula la dirección de salto cada vez que se ejecuta.
Se empieza por llenar el array de código y pokear directamente la dirección de la subrutina en él la dirección de salto calculada. Después se pokean la dirección y la longitud del nombre y el número del drive en las variables del sistema D_STR1, T_STR1 y N_STR1. Al final, se ejecuta el código contenido en el array.
Los parámetros utilizados son: NOMBRE: FILENAME y DRIVRE: DRIVENUM.

Procedimiento CAT (DRIVE, CANAL):

Cataloga el Microdrive DRIVE en la corriente CANAL.
Funciona de la misma forma que el FORMAT, por lo que no describiré el sistema de almacenar el código máquina.

LISTADO 1

Rutinas de acceso a ficheros secuencias en Pascal. El compilador utilizado es el Hisfot HP4TM161, pero cualquiera que soporte los procedimientos INLINE, PEEK, POKE y USER es válido.

```
  10 (* RUTINAS DE ACCESO A FIC
HEROS SECUENCIALES
  20
  30    COMPILADOR HISOFT HP4TM
161
  40
  50        Autor S. Casablancas
  60
  70 INSERTAR ANTES EN EL CUERP
O DE DECLARACIONES
  80
  90
  100
  110 CONST
  120   DSTR1 = 23766;
  130   DSTR11 = 23767;
  140   SSTR1 = 23768;
  150   LSTR1 = 23769;
  160   NSTR1 = 23770;
  170   TSTR1 = 23772;
  180   CHANS = 23631;
  190   STRMSR = ^5C16;
  200   CURCHL = 23633;
  210
  220
  230
  240 TYPE
  250   FILENAME=ARRAY[1..10] OF
 CHAR;
  260   DRIVENUM=1..8;
  270   CHANEL = 0..15;
  280     FILE = CULQUIER TIPO L
EGAL
  290
  300
  310
  320
  330
  340 FIN DE LAS DECLARACIONES *
)
  350
  360
  370 PROCEDURE ERROR (NUM,LOC:I
NTEGER ); FORWARD;
  380
  390
  400
  410 PROCEDURE OPEN (NOM:FILENA
ME;CANAL:CHANEL;DRIVE:DRIVENUM);
  420 VAR DESP,AUX:INTEGER;
  430 BEGIN
  440 AUX:= 2*CANAL+STRMSR;
  450 IF NOT (PEEK (AUX,INTEGER)
= 0)THEN ERROR (1,1);
  460 POKE (DSTR1,CHR(DRIVE));
  470 POKE (DSTR11,CHR(0));
  480 POKE (LSTR1,'M');
  490 POKE (NSTR1,SIZE(NOM));
  500 POKE (TSTR1,ADDR(NOM));
  510 INLINE (^FD,^21,^3A,^5C,^D
9,^E5,^D9,^DD,^E5,^CF,^22,^E5,^D
D,^CB,4,^BE,
  520    ^AF,^CF,^21,^E1,^DD,^E1,
^DD,^74,^FB,^DD,^75,^FA,^D9,^E1,
^D9);
  530 POKE (AUX,DESP);
  540 END; (*DE OPEN*)
  550
  560
  570
  580
  590
  600 FUNCTION LECOESC(CANAL:CHA
NEL):BOOLEAN;
  610 VAR BASE,AUX:INTEGER;
  620 BEGIN
  630 BASE:= 2*CANAL+STRMSR;
  640 AUX:= PEEK(CHANS,INTEGER)+
PEEK(BASE,INTEGER)-1;
  650 IF (PEEK(BASE,INTEGER)=0)
THEN ERROR (2,2);
  660 IF (PEEK(AUX+4,CHAR)<>'M')
 THEN ERROR (3,2);
  670 LECOESC:= ODD(ORD(PEEK(AUX
+24,CHAR)));
  680 END; (*DE LECOESC*)
  690
  700
  710
  720
  730
  740 FUNCTION CIN(CANAL:CHANEL)
:CHAR;
  750 VAR AUX:CHAR;
  760 BEGIN
  770 INLINE (^FD,^21,^3A,^5C,^D
D,^E5,^DD,^7E,2,^CD,1,^16,^CD,^E
6,^15,^F5,^3E,2,
  780    ^CD,1,^16,^F1,^DD,^E1,^D
D,^77,^FB);
  790 CIN:=AUX;
  800 END; (*DE CIN*)
  810
  820
  830
  840
  850
  860 PROCEDURE COUT (CANAL:CHAN
EL;CAR:CHAR);
  870 VAR AUX,BASE,A1:INTEGER;
  880   PROCEDURE SPOUT(C:CHAR);
  890       BEGIN
  900       INLINE (^FD,^21,^3A,
^5C,^DD,^7E,2,^D7);
  910       END;(*DE SPOUT*)
  920 BEGIN
  930 BASE:= PEEK(2*CANAL+STRMSR
,INTEGER);
  940 BASE:= BASE+PEEK (CHANS,IN
TEGER)-1;
  950 A1:= PEEK(CURCHL,INTEGER);
  960 POKE (CURCHL,BASE);
  970 SPOUT(CAR);
  980 POKE (CURCHL,A1);
  990 END: (*DE COUT*)
 1000
 1010
 1020
 1030
 1040
 1050 FUNCTION EOF (CANAL:CHANEL
):BOOLEAN;
 1060 VAR A:CHAR;
 1070 AUX,BASE:INTEGER;
 1080 BEGIN
 1090 BASE:= PEEK (2*CANAL+STRMS
R,INTEGER);
 1100 IF (BASE=0) THEN ERROR (2,
3);
 1110 BASE:= BASE+PEEK(CHANS,INT
EGER)-1;
 1120 IF (PEEK(BASE+4,CHAR)<>'M'
) THEN ERROR (3,3);
 1130 A:= PEEK (BASE+67,CHAR);
 1140 INLINE (^DD,^7E,^FB,^E6,2,
^DD,^77,^FB);
 1150 EOF:= ((A <> CHR (0)) AND
(PEEK(BASE+11,INTEGER)=PEEK(BASE
+69,INTEGER)));
 1160 END; (*DE EOF*)
 1170
 1180
 1190
 1200
 1210
 1220 PROCEDURE CLOSE (CANAL:CHA
NEL);
 1230 VAR AUX,BASE:INTEGER;
 1240 BEGIN
 1250 BASE:=PEEK(2*CANAL+STRMSR,
INTEGER);
 1260 IF (BASE=0) THEN  ERROR (2
,4);
 1270 AUX:=BASE+PEEK (CHANS,INTE
GER)-1;
 1280 IF (PEEK(AUX+4,CHAR)<>'M')
 THEN ERROR (3,4);
 1290 INLINE (^FD,^21,^3A,^5C,^D
D,^E5,^DD,^66,^FB,^DD,^6E,^FA,^E
5,^DD,^E1,
 1300    ^FD,^CB,^7C,^BE,^CF,^23,
^FD,^36,^7C,0,^DD,^E1);
 1310 POKE (BASE,0);
 1320 END; (* DE CLOSE *)
 1330
 1340
 1350
 1360
 1370
 1380 PROCEDURE ERASE (NOM:FILEN
AME; DRIVE:DRIVENUM);
 1390 BEGIN
 1400 POKE (DSTR1,CHR(DRIVE));
 1410 POKE (DSTR11,CHR(0));
 1420 POKE (LSTR1,'M');
 1430 POKE (NSTR1,SIZE(NOM));
 1440 POKE (TSTR1,ADDR(NOM));
 1450 INLINE (^FD,^21,^3A,^5C,^D
9,^E5,^D9,^DD,^E5,^CF,^24,^DD,^E
1,^D9,^E1,^D9);
 1460 END; (* DE ERASE*)
 1470
 1480
 1490
 1500
 1510
 1520 PROCEDURE RESET (NOM:FILEN
AME;CANAL:CHANEL;DRIVE:DRIVENUM)
;
 1530 BEGIN
 1540 IF (PEEK(2*CANAL+STRMSR,IN
TEGER)<>0) THEN
 1550   BEGIN
 1560   IF (PEEK (PEEK(2*CANAL+S
TRMSR,INTEGER)+PEEK(CHANS,INTEGE
R)+3,CHAR)<>'M')
 1570      THEN ERROR (3,5);
 1580   CLOSE (CANAL);
 1590   END; (*1 THEN*)
 1600 OPEN (NOM,CANAL,DRIVE);
 1610 IF LECOESC (CANAL) THEN ER
ROR (5,5);
 1620 END; (*DE RESET*)
 1630
 1640
 1650
 1660
 1670
 1680 PROCEDURE REWRITE (NOM:FIL
ENAME; CANAL:CHANEL; DRIVE:DRIVE
NUM);
```

```
1690 LABEL 1;
1700 BEGIN
1710 IF (PEEK (CANAL*2+STRMSR,INTEGER)<>0) THEN
1720   BEGIN
1730   IF (PEEK(PEEK(CANAL*2+STRMSR,INTEGER)+PEEK(CHANS,INTEGER)+3,CHAR)<>'M')
1740         THEN ERROR (3,6);
1750   CLOSE (CANAL);
1760   ERASE (NOM,DRIVE);
1770   END; (* 1 THEN*)
1780 1:OPEN (NOM,CANAL,DRIVE);
1790 IF NOT LECOESC (CANAL) THEN
1800   BEGIN
1810   CLOSE (CANAL);
1820   ERASE (NOM,DRIVE);
1830   GOTO 1;
1840   END; (* DE THEN*)
1850 END; (*DE REWRITE*)
1860
1870
1880
1890
1900
1910 PROCEDURE WRITEF (CANAL:CHANEL; FVAR:FILE);
1920 VAR BASE:INTEGER;
1930 BEGIN
1940 IF NOT LECOESC (CANAL) THEN ERROR (7,7);
1950 BASE:=PEEK(CANAL*2+STRMSR,INTEGER);
1960 IF (BASE=0) THEN ERROR (2,7);
1970 BASE:= BASE+PEEK (CHANS,INTEGER)-1;
1980 IF (PEEK (BASE+4,CHAR)<>'M') THEN ERROR (3,7);
1990 FOR BASE:= ADDR (FVAR) TO ADDR(FVAR)+SIZE(FVAR) DO
2000 COUT (CANAL,PEEK (BASE,CHAR));
2010 COUT (CANAL,CHR(13));
2020 END; (*DE WRITEF*)
2080
2090
2100
2110
2120
2130 PROCEDURE READF (CANAL:CHANEL; VAR FVAR:FILE);
2140 VAR BASE:INTEGER;
2150    A:CHAR;
2160 BEGIN
2170 IF LECOESC (CANAL) THEN ERROR (6,8);
2180 BASE:= PEEK (2*CANAL+STRMSR,INTEGER);
2190 IF (BASE=0) THEN ERROR (2,8);
2200 IF (PEEK(BASE+PEEK(CHANS,INTEGER)+3,CHAR)<>'M') THEN ERROR (3,8);
2210 FOR BASE:= ADDR (FVAR) TO ADDR (FVAR)+SIZE (FVAR) DO
2220 POKE (BASE,CIN (CANAL));
2230 A:= CIN (CANAL);
2240 END; (*DE READF*)
2250
2260
2270
2280
2290
2300 PROCEDURE FORMAT (NOM:FILENAME; DRIVE:DRIVENUM );
2310 VAR    CODE : ARRAY [1..33] OF CHAR;
2320 BEGIN
2330    CODE[1]:= CHR(^21); CODE[4]:= CHR(^FD);
2340    CODE[5]:= CHR(^21); CODE[6]:= CHR(^3A);
2350    CODE[7]:= CHR(^5C); CODE[8]:= CHR(^DD);
2360    CODE[9]:= CHR(^E5); CODE[10]:= CHR(^D9);
2370    CODE[11]:= CHR(^E5); CODE[12]:= CHR(^D9);
2380    CODE[13]:= CHR(^ED); CODE[14]:= CHR(^63);
2390    CODE[15]:= CHR(^ED); CODE[16]:= CHR(^5C);
2400    CODE[17]:= CHR(^CF); CODE[18]:= CHR(^32);
2410    CODE[19]:= CHR(^D9); CODE[20]:= CHR(^E1);
2420    CODE[21]:= CHR(^D9); CODE[22]:= CHR(^DD);
2430    CODE[23]:= CHR(^E1); CODE[24]:= CHR(^C9);
2440    CODE[25]:= CHR(^2A); CODE[26]:= CHR(^EB);
2450    CODE[27]:= CHR(^04); CODE[28]:= CHR(^23);
2460    CODE[29]:= CHR(^5E); CODE[30]:= CHR(^23);
2470    CODE[31]:= CHR(^56); CODE[32]:= CHR(^EB);
2480    CODE[33]:= CHR(^E9);
2490 POKE (ADDR(CODE[2]),ADDR(CODE[25]));
2500 POKE (DSTR1,CHR(DRIVE)); POKE(DSTR11,CHR(0));
2510 POKE (LSTR1,'M'); POKE(NSTR1,SIZE(NOM));
2520 POKE (TSTR1,ADDR(NOM)); USER (ADDR(CODE[1]));
2530 END; (* DE FORMAT*)
2540
2550
2560
2570
2580
2590 PROCEDURE CAT (DRIVE:DRIVENUM; CANAL:CHANEL);
2600 VAR    CODE : ARRAY [1..33] OF CHAR;
2610 BEGIN
2620    CODE[1]:= CHR(^21); CODE[4]:= CHR(^FD);
2630    CODE[5]:= CHR(^21); CODE[6]:= CHR(^3A);
2640    CODE[7]:= CHR(^5C); CODE[8]:= CHR(^DD);
2650    CODE[9]:= CHR(^E5); CODE[10]:= CHR(^D9);
2660    CODE[11]:= CHR(^E5); CODE[12]:= CHR(^D9);
2670    CODE[13]:= CHR(^ED); CODE[14]:= CHR(^63);
2680    CODE[15]:= CHR(^ED); CODE[16]:= CHR(^5C);
2690    CODE[17]:= CHR(^CF); CODE[18]:= CHR(^32);
2700    CODE[19]:= CHR(^D9); CODE[20]:= CHR(^E1);
2710    CODE[21]:= CHR(^D9); CODE[22]:= CHR(^DD);
2720    CODE[23]:= CHR(^E1); CODE[24]:= CHR(^C9);
2730    CODE[25]:= CHR(^2A); CODE[26]:= CHR(^B0);
2740    CODE[27]:= CHR(^04); CODE[28]:= CHR(^23);
2750    CODE[29]:= CHR(^5E); CODE[30]:= CHR(^23);
2760    CODE[31]:= CHR(^56); CODE[32]:= CHR(^EB);
2770    CODE[33]:= CHR(^E9);
2780 POKE (ADDR(CODE[2]),ADDR(CODE[25]));
2790 POKE (DSTR1,CHR(DRIVE)); POKE(DSTR11,CHR(0));
2800 POKE (SSTR1,CHR(CANAL)); POKE (LSTR1,'M');
2810 USER (ADDR(CODE[1]));
2820 END; (* DE CAT*)
2830
2840
2850
2860
2870
2880 PROCEDURE APPEND (NOM:FILENAME; CANAL:CHANEL; DRIVE:DRIVENUM);
2890 VAR BASE:INTEGER;
2900 BEGIN
2910 BASE:= PEEK (2*CANAL+STRMSR,INTEGER);
2920 IF (BASE =0) THEN OPEN (NOM,CANAL,DRIVE);
2930 BASE:= PEEK (2*CANAL+STRMSR,INTEGER)+PEEK (CHANS,INTEGER)-1;
2940 IF (PEEK(BASE+4,CHAR) <> 'M') THEN ERROR (3,9);
2950 POKE (SSTR1,CHR(CANAL));
2960 INLINE (^FD,^21,^3A,^5C,^DD,^E5,^D9,^E5,^D9,^3A,^D8,^5C,
2970         ^CD,1,^16,^DD,^2A,^51,^5C,^DD,^CB,^18,^46,
2980         ^20,^57,^DD,^CB,^43,^4E,^20,4,^CF,^25,^18,^F6,^AF,
2990         ^CF,^21,^DD,^6E,^45,^DD,^66,^46,^DD,^75,^B,^DD,^74,
3000         ^C,^11,0,2,^E5,^A7,^ED,^52,^28,^10,^DD,^36,^B,0,
3010         ^DD,^36,^C,0,^DD,^36,^45,0,^DD,^36,^46,0,^DD,^CB,
3020         ^43,^8E,^DD,^7E,^19,^CF,^21,^DD,^7E,^29,B,^DD,^7E,
3030         ^D,^F5,8,^DD,^77,^D,^CF,^2A,^AF,^CF,^21,^F1,^DD,
3040         ^77,^D,^E1,^DD,^75,^B,^DD,^74,^C,^DD,^CB,^18,^C6,
3050         ^D9,^E1,^D9,^DD,^E1);
3060 END; (* DE APPEND*)
3070
3080
3090
3100
3110
3120 PROCEDURE ERROR ; (* DECLARADA FORWARD EN PRINCIPIO *)
3130 BEGIN
3140 PAGE;
3150 WRITELN;
3160 CASE NUM OF
3170    1:WRITE ('ERROR:CORRENTE YA ABIERTA EN ');
3180    2:WRITE ('ERROR:CORRIENTE NO ABIERTA EN ');
3190    3:WRITE ('ERROR:CANAL NO MICRODRIVE EN ');
3200    4:WRITE ('ERROR:CANAL NO ACCESIBLE EN ');
3210    5:WRITE ('ERROR:FICHERO NO ENCONTRADO EN ');
3220    6:WRITE ('ERROR:FICHERO DE ESCRITURA EN ');
3230    7:WRITE ('ERROR:FICCHERO DE LECTURA EN ')
3240 END; (* DE CASE *)
3250 CASE LOC OF
3260    1:WRITE ('OPEN ');
3270    2:WRITE ('LECOESC ');
3280    3:WRITE ('EOF ');
3290    4:WRITE ('CLOSE ');
3300    5:WRITE ('RESET ');
3310    6:WRITE ('REWRITE ');
3320    7:WRITE ('WRITEF ');
3330    8:WRITE ('READF ');
3340    9:WRITE ('APPEND ')
3350 END; (* DE CASE *)
3360 WRITELN;
3370 HALT;
3380 END; (* DE ERROR *)
```

El compilador utiliza el registro IY, por lo que debe ser devuelto a su valor original antes de intentar cualquier llamada a la ROM

Se pokean las variables del sistema D_STR1 y S_STR1 los valores del drive a catalogar y el canal de salida y se ejecuta el código contenido en el array.

No debe estar abierto ningún canal de Microdrive al invocar CAT. Las corrientes válidas son 1, 2 (pantalla) y 3 (impresora).

Procedimiento ERROR (NUM. LOC):

Es el encargado de mostrar el tipo y la localización de los errores en tiempo de ejecución.

Los parámetros son del tipo INTEGER, y contienen NUM el tipo de error y LOC la sentencia donde se ha producido.

Se declara FORWARD en el principio para que pueda ser utilizada por las otras rutinas, pero se escribe al final para que pueda ser alargada por el usuario para añadir más tipos de error.

La explicación de las rutinas en código máquina, requiere una previa explicación de cómo trabaja el compilador en el almacenamiento de variables.

Las variaciones globales, se guardan en la parte alta de la pila de ejecución y hacia abajo, en el mismo orden en que son declaradas. Ocupan tantos *bytes* como el total de sus componentes.

Las variables locales son direccionadas mediante el registro IX, que está asignado al principio de cada bloque interno. Por tanto, IX-4 apunta al principio del bloque de variables. Estas se encuentran desde aquí hacia abajo en orden de declaración. Por ejemplo, si la primera variable declarada es AUX: INTEGER; el *byte* alto se encontrará en (IX-5) y el bajo en (IX-6).

Los parámetros valor son direccionados también por el registro IX, pero hacia arriba. Cuando antes es declarado, más alta en la memoria estará su posición. La dirección más baja está en (IX+2).

Los parámetros variables son tratados como los parámetros valor, pero siempre tienen asignados 2 *bytes* que contienen su dirección en la pila de ejecución.

Los valores devueltos se sitúan a continuación de los parámetros (valor o variable). También direccionados mediante IX.

Las funciones predefinidas ADDR (v) y SIZE (v), dan valores correctores en todos los casos excepto en los valores devueltos.

La llamada al código máquina se puede realizar de dos formas, mediante un procedimiento USER (dirección), que llama a una rutina fija en la memoria acabada en RETURN, y INLINE (código), que incluye el código en el objeto generado por el compilador.

Debido al funcionamiento del sistema operativo del Spectrum, debe tenerse cuidado con el contenido del registro IY, utilizado para direccionar las variables del sistema. El compilador lo utiliza, por lo que debe ser puesto a su valor adecuado antes de intentar ejecutar cualquier llamada a la ROM.

Estas rutinas funcionan con cualquier versión de la ROM del Interface 1, que aprovecha que el analizador sintáctico está localizado en la misma posición en ambas versiones. Las direcciones absolutas de algunas rutinas se extraen del analizador sintáctico una vez paginada la ROM.

Debido a la poca memoria que queda libre por debajo del Ejecutor, solamente se puede tener abierto un canal de microdrive a la vez. Se debe tener cuidado con las instrucciones CAT, TOUT, TIN o FORMAT porque utilizan un canal M y si ya hay uno en existencia puede bloquearse la máquina.

Los datos se graban con igual representación que la que tienen en la memoria, seguidos de un caracter CR (chr(13)) para compatibilizarlos con el BASIC. Los enteros ocupando dos *bytes*, los reales cinco *bytes* y los registros, arrays, etc. la suma total del tamaño de sus componentes. La separación entre campos de los registros, es posicional en el orden en que se han declarado.

Los enteros se graban en forma de caracteres según formato Intel (lsb, msb), y los reales, en caracteres según formato de coma flotante propia del Spectrum. Los caracteres, se graban en ASCII y el tipo BOOLEAN, en forma chr (1) para TRUE y CHR (0) para FALSE.

Los procedimientos que puedan modificar el archivo en el Microdrive, REWRITE, ERASE, FORMAT, WRITEF..., comprueban la protección contra escritura, y en caso de que el Microdrive esté protegido, para el proceso con mensaje de error System Call Error.

Como dato curioso, cabe destacar que los archivos creados con el procedimiento TOUT, son del tipo PRINT, y se pueden leer mediante estas rutinas.

Santi Casablancas

LISTADO 2

Rutinas en Assembler. Se proporciona únicamente como documentación, pues el código objeto está incluido en el programa Pascal.

```
   1 ;RUTINAS EN ASSEMBLER
   2
   3
   4 ;RUTINA DE OPEN
   5
   6 OPEN    LD    IY,^5C3A ; PON
ER IY EN VARIABLES DEL SISTEMA
   7
   8         EXX
   9         PUSH HL         ;SALV
AR REGISTROS
  10         EXX
  11         PUSH IX
  12         RST  ^8
  13         DEFB ^22        ;ABRI
R CANAL
  14         PUSH HL         ;GUAR
DAR DESPLAZAMIENTO
  15         XOR  A
  16         RST  ^8
  17         DEFB ^21        ;PARA
R MOTOR
  18         RES  7,(IX+4) ;HACE
R CANAL PERMANENTE
  19         POP  HL
  20         POP  IX         ;RECU
PERAR DES. Y POINTER PASCAL
  21         LD   (IX-6),L ;PONE
R DESP. EN VARIABLE PASCAL
  22         LD   (IX-5),H
  23         EXX
  24         POP  HL
  25         EXX  ;RECUPERAR REG
ISTROS
  26 ; NO HAY RETURN: ES PROCED
IMIENTO INLINE
  27
  28
  29
  30 ;RUTINA DE CLOSE
  31
  32 CLOSE  LD    IY,^5C3A ; PON
ER IY EN VARIABLES SISTEMA
  33         PUSH IX ; SALVAR PU
NTERO PASACL
  34         LD   H,(IX-5) ; CAR
GAR DIR. CANAL DESDE VARIABLE PA
SCAL
  35         LD   L,(IX-6)
  36         PUSH HL
  37         POP  IX ; CARGARLO
EN IX
  38         RES  1,(IY+124) ; I
NDICAR QUE NO ES CLEAR^
  39         RST  ^8
```

```
        DEFB ^23 ; CERRAR CANAL
        LD   (IY+124),0 ; PONER 0 EN FLAGS 3
        POP  IX
; NO HAY RETUR: ES PROCEDIMIENTO INLINE

;RUTINA DE INPUT

CIN     LD   IY,^5C3A ; PONER IY EN VARIABLES DEL SISTEMA
        PUSH IX ; SALVAR PUNTERO PASCAL
        LD   A,(IX+2) ; CARGAR NUMERO CORRIENTE DESDE PASCAL
        CALL ^1601 ; ABRIR CORRIENTE
        CALL ^15E6 ; ENTRAR UN CARACTER
        PUSH AF ; GUARDAR CARACTER
        LD   A,2 ; ABRIR CORRIENTE 2
        CALL ^1601
        POP  AF
        POP  IX ; RECUPERAR CARACTER Y PUNTERO
        LD   (IX-5),A ; PONER CARACTER EN VARIABLE PASCAL
; NO HAY RETURN: PROCEDIMIENTO INLINE

;RUTINA DE OUTPUT

COUT    LD   IY,^5C3A ; PONER IY EN VARIABLES DEL SISTEMA
        LD   A,(IX+2) ; CARGAR A CON CARACTER A SACAR
        RST  ^10 ; LLAMAR A OUTPUT DE LA ROM
; NO HAY RETURN: PROCEDIMIENTO INLINE

;RUTINA DE APPEND

APPEND  LD   IY,^5C3A ; PONER IY EN VARIABLES DEL SISTEMA
        PUSH IX ; SALVAR REGISTROS
        EXX
        PUSH HL
        EXX
        LD   A,(23768) ; CARGAR NUM.CORRIENTE
        CALL ^1601
        LD   IX,(23633) ; PONER DIRECCION EN IX
        BIT  0,(IX+24) ; MIRAR ESCRITURA
        JR   NZ,FIN ; SI ES C. PASAR AL FINAL
BUCLE   BIT  1,(IX+67) ; MIRAR EOF
        JR   Z,FINBUC ; SI EOF ACABAR BUCLE
        RST  ^8
        DEFB ^25 ; LEER SECTOR SIGUIENTE
        JR   BUCLE ; REPETIR
FINBUC  XOR  A
        RST  ^8
        DEFB ^21 ; PARAR MOTOR
        LD   L,(IX+69)
        LD   H,(IX+70) ; CARGAR NUM. BYTES QUE CONTIENE
        LD   (IX+11),L
        LD   (IX+12),H ; HACER APUNTADOR = FINAL REG.
        LD   DE,512 ; MIRAR SI FINAL = 512
        PUSH HL ; SALVAR APUNTADOR
        AND  A
        SBC  HL,DE ; COMPROVARLO
        JR   Z,PLE ; SALTAR ESTO SI LLENO
        LD   (IX+11),0
        LD   (IX+12),0
        LD   (IX+69),0 ; PREPARAR BORRADO DE SECTOR
        LD   (IX+70),0
PLE     RES  1,(IX+67) ; QUITAR FLAG DE EOF
        LD   A,(IX+25) ; BUSCAR NUM. MICRODRIVE
        RST  ^8
        DEFB ^21 ; PONER MOTOR EN MARCHA
        LD   A,(IX+41) ; BUSCAR NUM. SECT. FISICO
        EX   AF,AF' ; GUARDARLO
        LD   A,(IX+13) ; BUSCAR NUM. SECT. LOGICO
        PUSH AF ; GUARDARLO
        EX   AF,AF' ; RECUPERAR SEC. FISICO
        LD   (IX+13),A ; PONERLO EN APUNT.
        RST  ^8
        DEFB ^2A ; REESCRIBIR SECTOR
        XOR  A
        RST  ^8
        DEFB ^21 ; PARAR MOTOR
        POP  AF ; RECUPERAR SECTOR LOGICO
        LD   (IX+13),A ; PONERLO EN SU SITIO
        POP  HL ; RECUPERAR CANTIDAD BYTES
        LD   (IX+11),L
        LD   (IX+12),H ; PONER EN SU SITIO
FIN     SET  0,(IX+24) ; PONER 0 EN FLAGS3
        EXX
        POP  HL
        EXX
        POP  IX ; RECUPERAR REGISTROS
; NO HAY RETURN: PROCEDIMIENTO INLINE

;RUTINA DE ERASE

ERASE   LD   IY,^5C3A
        EXX  ; PONER IY EN VARIABLES DEL SISTEMA
        PUSH HL
        EXX
        PUSH IX ; SALVAR REGISTROS
        RST  ^8
        DEFB ^24 ; BORRAR ARCHIVO
        POP  IX
        EXX
        POP  HL ; RECUPERAR REGISTROS
        EXX
; NO HAY RETURN: PROCEDIMIENTO INLINE

;RUTINA DE FORMAT

FORMAT  DEFB ^21 ; PRIMERBIT INSTRUCC. LD HL
        DEFS 2 ; ESPACIO PARA RELOCALIZACION
        LD   IY,^5C3A ; PONER IY EN VARIABLES DEL SISTEMA
        PUSH IX
        EXX
        PUSH HL
        EXX  ; SALVAR REGISTROS
        LD   (^5CED),HL ; CARGAR HD_11 CON DIRECCION FORM1 CALCULADA
        RST  ^8
        DEFB ^32 ; PAGINAR ROM Y SALTAR A SUBRUTINA FORM1
        EXX
        POP  HL
        EXX
        POP  IX ; RECUPERAR REGISTROS
        RET  ; ESTA RUTINA ESTA LOCALIZADA EN UN ARRAY
FORM1   LD   HL,(^4EB) ; DIRECCION ANALIZADOR SINTACTICO DE FORMAT
        INC  HL
        LD   E,(HL)
        INC  HL
        LD   D,(HL) ; BUSCAR DIRECCION EN ROM BK
        EX   DE,HL
        JP   (HL) ; EJECUTAR FORMAT

;RUTINA DE CAT

CAT     DEFB ^21 ; PRIMERBIT INSTRUCC. LD HL
        DEFS 2 ; ESPACIO PARA RELOCALIZACION
        LD   IY,^5C3A ; PONER IY EN VARIABLES DEL SISTEMA
        PUSH IX
        EXX
        PUSH HL
        EXX  ; SALVAR REGISTROS
        LD   (^5CED),HL ; CARGAR HD_11 CON DIRECCION CAT1 CALCULADA
        RST  ^8
        DEFB ^32 ; PAGINAR ROM Y SALTAR A SUBRUTINA CAT1
        EXX
        POP  HL
        EXX
        POP  IX ; RECUPERAR REGISTROS
        RET  ; ESTA RUTINA ESTA LOCALIZADA EN UN ARRAY
CAT1    LD   HL,(^4EB) ; DIRECCION ANALIZADOR SINTACTICO DE CAT
        INC  HL
        LD   E,(HL)
        INC  HL
        LD   D,(HL) ; BUSCAR DIRECCION EN ROM BK
        EX   DE,HL
        JP   (HL) ; EJECUTAR CAT
```

AMSTRAD CPC - 464 AMSTRAD



ORDENADORI

SERIE CPC

UNIDAD CENTRAL. MEMORIAS
• Microprocesador Z80A - 64K RAM ampliables - 32K ROM ampliables
• **TECLADO** • Teclado profesional con 74 teclas en 3 bloques - Hasta 32 teclas programables - Teclado redefinible
• **PANTALLA** • Monitor RGB verde (12") o color (14")

	Normal	Alta Res.	Multicolor
Col × líneas	40 × 25	80 × 25	20 × 25
Colores	4 de 27	2 de 27	16 de 27
Puntos	320 × 200	640 × 200	160 × 2

— Se pueden definir hasta 8 ventanas de texto y 1 de gráficos • **SONIDO**
• 3 canales de 8 octavas moduladas independientemente - Altavoz interno regulable - Salida estéreo • **BASIC**
• Locomotive BASIC ampliado en ROM - Incluye los comandos AFTER y EVERY para control de interrupciones

AMSTRAD CPC 464

CASSETTE • Cassette incorporada con velocidad de grabación (1 ó 2 Kbaudios) controlada desde Basic • **CONECTORES**
• Bus PCB multiuso, Unidad de Disco exterior, paralelo Centronics, salida estéreo, joystick, lápiz óptico, etc.
• **SUMINISTRO** • Ordenador con monitor verde o color - 8 cassettes con programas - Libro "Guía de Referencia BASIC para el programador" - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	59.900 Pts. (monitor verde)
	90.900 Pts. (monitor color)

AMSTRAD CPC 6128

UNIDAD DE DISCO • Unidad incorporada para disco de 3" con 180K por cara • **SISTEMAS OPERATIVOS**
• AMSDOS, CP/M 2.2, CP/M Plus (3.0)
• **CONECTORES** • Bus PCB multiuso, paralelo Centronics, cassette exterior, 2.ª Unidad de Disco, salida estéreo, joysticks, lápiz óptico, etc.
• **SUMINISTRO** • Ordenador con monitor verde o color - Disco con CP/M 2.2 y lenguaje DR. LOGO - Disco con CP/M Plus y utilidades - Disco con 6 programas de obsequio - Manual en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR	84.900 Pts. (monitor verde)
	119.900 Pts. (monitor color)

PCW - 8256

AMSTRAD CPC - 6128

ES AMSTRAD ¡¡Increíble!!

AMSTRAD PCW 8256

UNIDAD CENTRAL. MEMORIAS • Microprocesador Z80A - 256K RAM de las que 112K se utilizan como disco RAM • **TECLADO** • Teclado profesional en castellano (ñ, acento...) de 82 teclas • **PANTALLA** • Monitor verde de alta resolución - 90 columnas × 32 líneas de texto • **UNIDAD DE DISCO** • Disco de 3" y 173K por cara - Opcionalmente, 2.ª Unidad de Disco de 1 Mbyte integrable • **SISTEMA OPERATIVO** • CP/M Plus de Digital Research • **IMPRESORA** • Alta calidad (NLQ) a 20 c.p.s. - Calidad estándar a 90 c.p.s. - Papel continuo u hojas sueltas - Alineación automática del papel - Caracteres normales, comprimidos, expandidos, control del paso de letra (normal, cursiva, negrita, subíndices, superíndices, subrayado, etc). • **OPCIONES** • Kit de Ampliación a 512K RAM y 2.ª Unidad de Disco - Interface Serie RS 232C y paralelo Centronics • **SUMINISTRO** • Ordenador completo con teclado, pantalla, Unidad de Disco e Impresora - Discos con el procesador de Texto LocoScript, CP/M Plus, Mallard, BASIC, DR. LOGO y diversas utilidades - Manuales en castellano - Garantía Oficial AMSTRAD ESPAÑA.

TODO POR 129.900 Pts.



Los mas prestigiosos paquetes de **Software Profesional**, en formato AMSTRAD... a **"precios AMSTRAD"**

Existe también la versión **PCW 8512** con **512K RAM** y la 2.ª Unidad de Disco de 1 Mbyte incorporada. **PVP. 174.900 Pts.**

* El **PCW 8256** puede utilizarse como terminal y en comunicaciones.

El I.V.A. no está incluido en los precios.

NOTA: Es muy importante verificar la garantía del aparato ya que sólo **AMSTRAD ESPAÑA** puede garantizarle la ordenada reparación y sobre todo materiales de repuesto oficiales (Monitor, ordenador, cassette o unidades de discos).

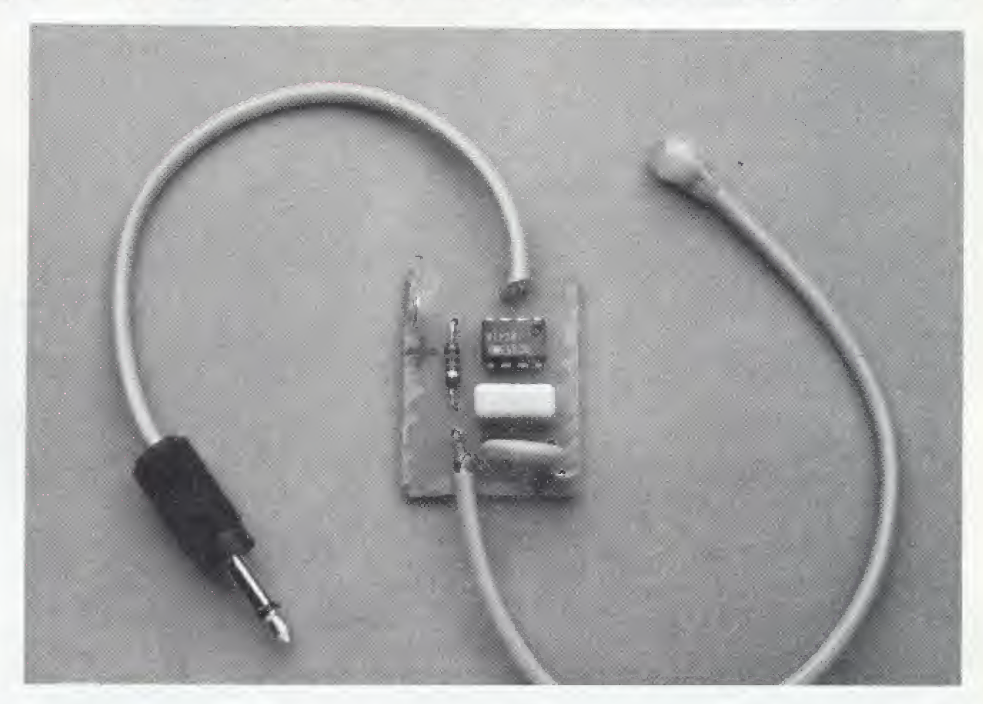
AMSTRAD ESPAÑA

Avda. del Mediterráneo, 9. Tels. 433 45 48 - 433 48 76. 28007 MADRID

Delegación Cataluña: Tarragona, 110 - Tel. 325 10 58. 08015 BARCELONA

TERMOMET

PARA SPECTRUM



Cuando alguien hace esa pregunta tan frecuente de ¿para qué sirve un ordenador?, la respuesta debería ser que un ordenador es un ladrillo, que no sirve para gran cosa, pero es el elemento de construcción que nos permite realizar cualquier proyecto que nos imaginemos, con una versatilidad fantástica.

Y aquí tenemos uno de esos proyectos, un termómetro construido con un mínimo de componentes y que realizado con elementos discretos necesitaría docenas de ellos para las mismas prestaciones (presentación digital, memoria continua, etcétera).

Pero veamos el montaje, que sólo consta de cinco componentes: un circuito integrado archiconocido como es el 555, dos condensadores y dos resistencias, una de ellas, de tipo NTC, como elemento sensor de la temperatura.

Para los menos expertos diremos que una resistencia NTC (**negative temperature coefficient**) o termistor tiene la característica de disminuir su resistencia conforme aumenta su temperatura. Desgraciadamente esta variación no es lineal, sino logarítmica, por lo cual no son empleadas como transductores de temperatura en termómetros a pesar de ser muy baratos.

Afortunadamente, aquí interviene nuestro Spectrum, que con unos sencillos cálculos en programa nos «aplana» la respuesta de la NTC, permitiendo realizar un termómetro con una precisión muy aceptable, que en algunos casos puede ser de una décima de grado.

El funcionamiento del circuito no puede ser más simple, al utilizar el CI 555 como multivibrador, cuya frecuencia de salida está determinada por un condensador y por la NTC (ver esquema de fig. 1). Ya hemos comentado que la resistencia de la NTC varía con la temperatura, lo cual ocasionará una variación proporcional (pero logarítmica) de la frecuencia de salida en la patilla 3 del 555, y que se aplica a la entrada MIC del Spectrum. Se ha preferido esta entrada ya que si lo hacemos por la EAR, el sonido por el altavoz del ordenador es más fuerte y puede ser molesto en una utilización prolongada.

La NTC empleada es de 47.000 ohmios lo cual significa que tendrá aproximadamente este valor de resistencia a 25 grados centígrados. Esto tiene su importancia, como veremos a la hora del ajuste.

Para alojar los componentes se ha utilizado un minúsculo circuito impreso cuyo dibujo por ambas caras aparece en la figura 2.

El montaje se alimenta en los puntos señalados, con una pila o alimentador de 5 a 16 voltios, aunque los ajustes del presente montaje se han realizado con 9 V. El consumo del 555 es mínimo.

Terminado el montaje del circuito impreso con los componen-



tes, debemos fijar la NTC (del tamaño de una lenteja) al final de un cable apantallado, recortando sus patillas al mínimo y una vez soldada (sin cortocircuitos), recubrirla de Araldit rápido hasta formar una bola endurecida que permita la introducción en líquidos, evite cortocircuitos al tocarla con los dedos y le de mayor solidez mecánica. Este aislamiento puede mejorarse con un par de capas de esmalte de uñas (que añade una nota de bonito color).

Todos los periféricos necesitan un programa para funcionar. Este no podía ser menos y tiene dos, uno en BASIC y otro en código máquina.

El programa 2 es la versión en ensamblador de la parte de código. Se coloca en la dirección 60.000, pero puede ser relocalizado en cualquier otro sitio disponible. En la línea 20 y 30 se cargan dos contadores, el HL sirve como almacén de los impulsos recibidos del integrado 555 y que son leídos en el puerto 254 en las líneas 80 y 160, comprobando si hay señal en las líneas 90 y 170. Se usan dos bucles (LOOP1 y LOOP2) para detectar cuándo hay señal o no en el puerto. Por último, el resultado de la cuenta se guarda en el stack con PUSH HL y se pasa al registro BC en la línea 210. La razón de este último intercambio de registros, es que cuando en BASIC hacemos un:

LET variable = USR XXXXX

al regresar al BASIC la variable contiene el valor del registro BC.

El programa 1 es la parte de BA-

PROGRAMA 1

```
  10 REM JOAQUIN PAREDES PARDO
  20 FOR d=60000 TO 60033
  30 READ a: POKE d,a: NEXT d
  50 LET f=USR 60000
  55 LET rt1=(1.44/(2*f*.0000000
04))-(1200/2)
  58 LET t1=3920/(((3920/298)-LN
 47000)+LN rt1)
  65 LET c=INT ((t1-273)*10.6*10
)
  66 LET t=(INT (c/10))/10
  68 PRINT AT 8,16;"       "
  70 PRINT AT 8,3;"TEMPERATURA:
";t;AT 8,21;"grados"
  80 GO TO 50
 100 DATA 33,0,0,1,255,255,11,12
0,177,40,20,219,254,254,191,40,2
45,35,11,120,177,40,8,219,254,25
4,191,32,245,24,231,229,193,201
```

PROGRAMA 2

```
 10          ORG   60000
 20          LD    HL,00000
 30          LD    BC,65535
 40 LOOP1    DEC   BC
 50          LD    A,B
 60          OR    C
 70          JR    Z,FIN
 80          IN    A,(254)
 90          CP    191
100          JR    Z,LOOP1
110          INC   HL
120 LOOP2    DEC   BC
130          LD    A,B
140          OR    C
150          JR    Z,FIN
160          IN    A,(254)
170          CP    191
180          JR    NZ,LOOP2
190          JR    LOOP1
200 FIN      PUSH  HL
210          POP   BC
220          RET
```

PROGRAMA 3



```
   1 DIM v(49): LET x=0
   2 PRINT "Para ajustar hora pu
lse 'a'"
   3 GO SUB 8000
   4 BORDER 2: PAPER 6
   6 GO TO 85
  10 BEEP .1,20
  12 GO SUB 8030
  14 RETURN
  85 POKE 23672,0: POKE 23673,0:
 POKE 23674,0
  90 CLS
  95 PRINT AT 1,28;"seg": PRINT
AT 1,24;"min": PRINT AT 1,20;"ho
r"
 100 DEF FN d(a,b)=a*(a>b)+b*(b>
=a)
 110 DEF FN j()=INT ((65536*PEEK
 23674+256*PEEK 23673+PEEK 23672
)/50)
 120 DEF FN t()=FN d(FN j(),FN j
())
 140 GO TO 470
 205 LET p=FN t(): LET t=p
 210 LET p=FN t(): LET t=p
 220 LET hor=INT (t/3600)
 230 LET t=t-hor*3600
 240 LET min=INT (t/60)
 250 LET t=t-min*60
 260 LET seg=INT t
 270 IF INKEY$="a" THEN  GO SUB
1000
 280 LET h$=STR$ hor: LET m$=STR
$ min: LET s$=STR$ seg: LET u$=h
$+m$+s$
 460 RETURN
 470 GO SUB 200
 480 PRINT AT 3,29;seg: IF seg=0
 THEN  PRINT AT 3,30;" "
 490 PRINT AT 3,25;min: IF min=0
 THEN  PRINT AT 3,26;" "
 500 PRINT AT 3,21;hor: IF hor>2
3 THEN  GO TO 7
 600 IF seg=00 THEN  GO SUB 10
 999 GO TO 470
1000 INPUT "hora (hhmmss) ? ";a$
: IF LEN a$<>6 THEN  GO TO 1000
1010 LET x=50*(VAL a$(5 TO 6)+60
*VAL a$(3 TO 4)+3600*VAL a$(1 TO
 2))
1020 LET x1=INT (x/65536): LET x
=x-65536*x1: LET x2=INT (x/256):
 LET x=x-256*x2: LET x3=x
1030 POKE 23674,x1: POKE 23673,x
2: POKE 23672,x3
1040 RETURN
8000 REM termometro
8010 FOR d=60000 TO 60033
8020 READ a: POKE d,a: NEXT d
8025 RETURN
8030 LET f=USR 60000
8040 LET rt1=(1.44/(2*f*.0000000
04))-(1200/2)
8050 LET t1=3920/(((3920/298)-LN
 47000)+LN rt1)
8060 LET c=INT ((t1-273)*10.6*10
)
8070 LET t=(INT (c/10))/10
8080 PRINT AT 8,16;"      "
8085 PRINT AT 8,3;"TEMPERATURA:
";t;AT 8,21;"grados"
8086 PAUSE 50
8088 IF min=00 OR min=15 OR min=
30 OR min=45 THEN  GO SUB 9000
8090 RETURN
8100 DATA 33,0,0,1,255,255,11,12
0,177,40,20,219,254,254,191,40,2
45,35,11,120,177,40,8,219,254,25
4,191,32,245,24,231,229,193,201
9000 REM almacena
9010 LET x=x+1
9020 LET v(x)=t
9030 RETURN
9500 REM grafica
9505 LET t=0: LET d=0
9510 FOR f=0 TO 48
9515 LET d=d+1
9520 LET y=4*v(d)
9570 IF NOT t THEN  PLOT f,(y-4)
+88: LET t=1: GO TO 9600
9580 DRAW 4,y-oldy
9600 LET oldy=INT (y+.5)
9610 NEXT f
```

Todospectrum



TODOSPECTRUM es una publicación mensual que le ayudará a obtener el máximo partido a su SPECTRUM y al ZX 81.

CONOZCA LAS VENTAJAS DE SUSCRIBIRSE A

Todospectrum





ADEMAS, le hacemos un 25 % DE DESCUENTO

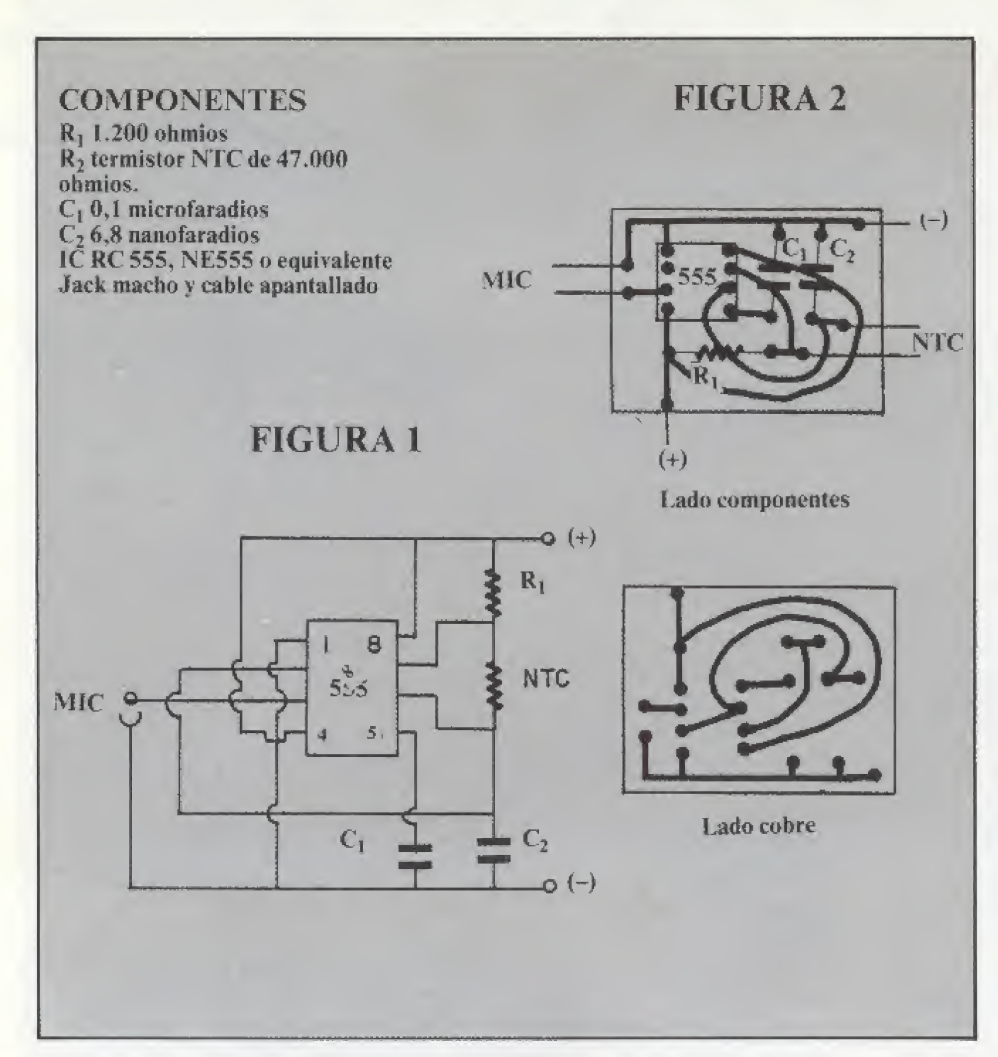
sobre el precio real de suscripción (12 números)

VALOR REAL DE SUSCRIPCION	OFERTA ESPECIAL DE SUSCRIPCION	USTED AHORRA
~~3.600~~ PTAS.	2.700 PTAS.	900 PTAS.

APROVECHE AHORA esta oportunidad irrepetible para suscribirse a **TODOSPECTRUM**. Envie **HOY MISMO** la tarjeta adjunta a la revista, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de **TODOSPECTRUM** más el **REGALO**.

Todospectrum

Bravo Murillo, 377
Tel. 733 79 69
28020 MADRID



SIC que se encarga de colocar el código máquina en su lugar, llamarlo para que realice la medida de la frecuencia generada por el montaje, tratar este valor con las fórmulas adecuadas para obtener finalmente la temperatura que corresponde a un determinado valor de resistencia de la NTC y finalmente imprimir en pantalla el resultado.

En la línea 55, a partir de la frecuencia medida mediante el código máquina hallamos la resistencia (rt1) de la NTC para la temperatura a la cual se encuentra. Seguidamente, usamos este valor (rt1) en la fórmula de la línea 58 para obtener la temperatura (t1), al mismo tiempo que pasamos los datos de su forma logarítmica a lineal.

Finalmente en las líneas 65 y 66 pasamos los grados de Kelvin a centígrados, ajustamos su valor y eliminamos los decimales innecesarios.

Llegados a este punto es necesario proceder al ajuste del dispositivo. Es imposible realizarlo de forma teórica usando para las fórmulas los valores de los componentes del montaje, entre otras cosas por-

que hay uno desconocido: la constante K de la NTC. Este es un valor asociado a cada tipo de NTC, pero que difícilmente podrán facilitarnos en el comercio suministrador (los componentes en España se venden como churros). En la línea 58 se ha asignado a K el valor 3920, que es más o menos típico. Por lo dicho procederemos de una forma más práctica y efectiva.

Los que han tenido la paciencia de leer hasta aquí recordarán que nuestra NTC tenía una resistencia de 47.000 a una temperatura de 25 grados centígrados. Así pues, prepararemos dos baños, uno a 25 y otro a 0 grados; este último se prepara con hielo de agua destilada picado. Unos cubitos flotando en agua darán 4 grados, no cero.

Una vez la sonda en agua a 25 grados (medir con un buen termómetro de mercurio) y estabilizada su temperatura, es evidente que rt1 en la línea 55 debería valer 47.000. Si no es así, debe modificarse el valor 000000004 en más o en menos hasta lograrlo. Para este ajuste puede escribirse temporalmente la línea 56 PRINT rt1. Esto puede ser suficiente para una precisión aceptable. En caso contrario, se prueba también a cero grados y se modifican los valores 1.6 ó 3920 de las líneas 65 y 58.

La gran ventaja de un termómetro controlado por ordenador es la continuidad y memorización de la medida, permitiendo su posterior procesado

De los tres prototipos construidos, dos necesitaron el ajuste anterior, mientras, el tercero funcionó a la primera con los valores del programa adjunto.

Evidentemente la precisión del dispositivo será mayor para márgenes cortos de medida (aprox. 0,1 grado) que para otros más amplios. Así, en el margen –10 a 40 grados puede esperarse una precisión de 0,5 a 1 grado (todo depende de la precisión del ajuste).

La gran ventaja de un termómetro a través de ordenador, aparte de su bajo precio en este caso, es la continuidad y memorización de la medida, lo que permite su posterior procesado en usos tan interesantes como, por ejemplo, el industrial o la experimentación (control automático de la temperatura de un animal ante un antipirético por ejemplo).

El programa 3 ha sido utilizado para memorizar la temperatura durante doce horas nocturnas y en períodos de 15 minutos (línea 8088); permitiendo la visualización en una gráfica con GO TO 9500 (no RUN). Evidentemente el programa admite muchas modificaciones y perfeccionamientos.

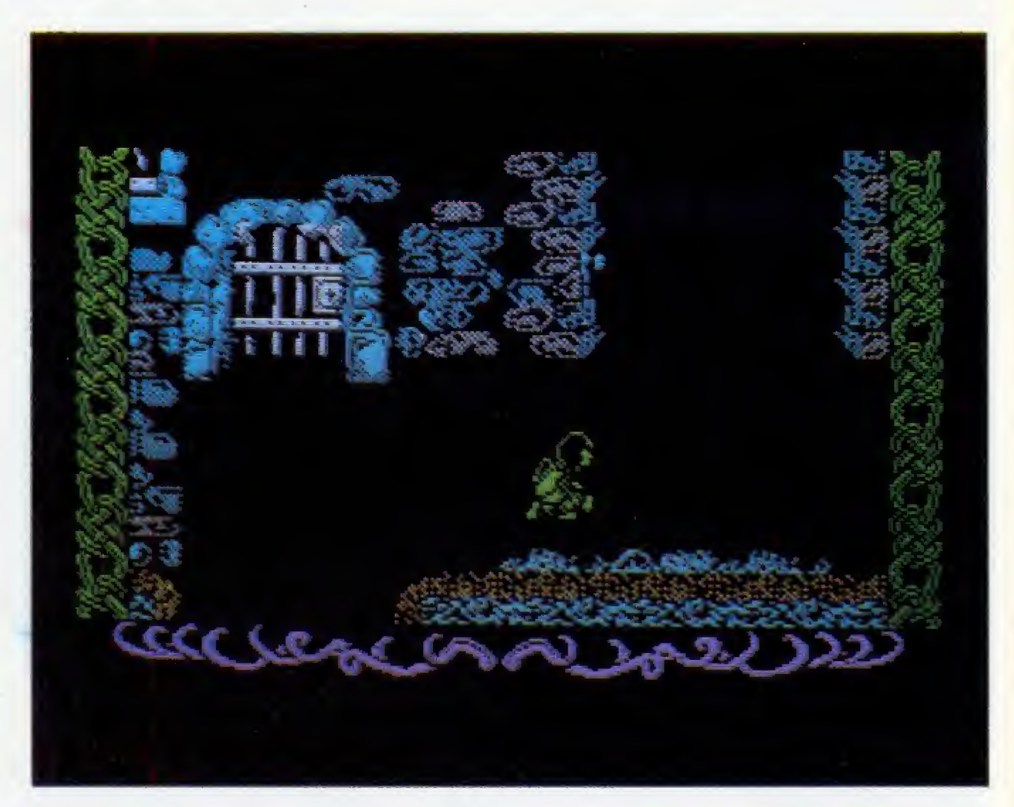
Asimismo, los más temerosos por la integridad de su querido Spectrum tras largas horas de funcionamiento continuo, pueden usar el programa para que el ordenador pida ¡socorro! (con programa sintetizador de voz) cuando el usuario crea que la temperatura de la máquina alcanza valores peligrosos para su bolsillo.

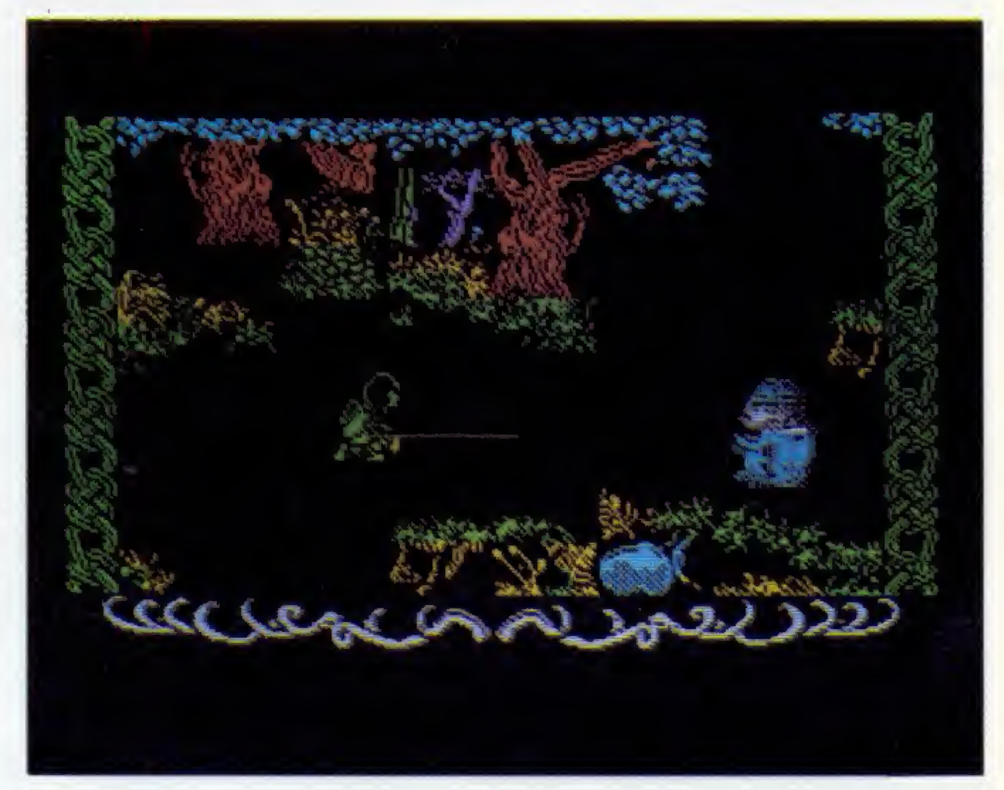
Joaquín Paredes

ROBIN OF THE WOOD

SERMA

SPECTRUM 48 K





Después del éxito obtenido con Nodes of Yesod la cada ODIN vuelve a la carga con un nuevo programa para Spectrum: Robin of the Wood.

Seguramente todos recordaréis la leyenda de Robín de los bosques, el legendario héroe medieval. Pues bien, con este magnífico programa tendremos la oportunidad de retroceder en el tiempo y ayudar a nuestro héroe a recuperar la gran flecha de plata, símbolo de libertad para los habitantes de Saxon. Esta flecha está en poder del malvado sherif de Nottingham, que ha organizado un gran torneo ofreciéndola como premio al vencedor. Pero todo esto no es más que una trampa para atrapar al valeroso Robin.

Tu misión consiste en llegar hasta el castillo de Nottingham y recuperar la preciada flecha.

Pero antes debes hacerte con tu arco, tu espada y tres flechas mágicas, tarea que no resulta demasiado fácil teniendo en cuenta que el bosque se encuentra lleno de soldados con una orden muy concreta: acabar contigo. También encontrarás otra serie de objetos, como coronas de laurel que aumentarán las vidas, flores, sacos de dinero y otros que serán de utilidad en determinadas ocasiones.

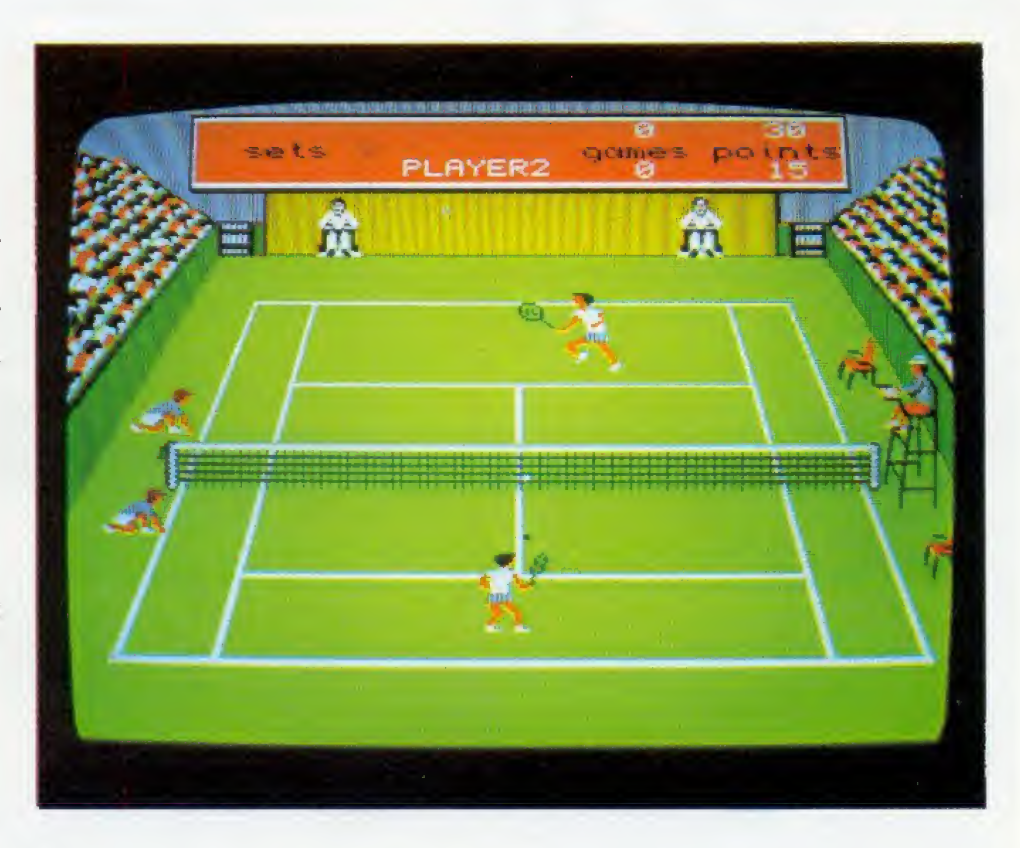
La difícil misión encomendada unido al gran número de pantallas, hacen de éste un juego difícil al que tendremos que dedicar horas y una gran dosis de paciencia si queremos llegar al final. Los gráficos son realmente buenos, el movimiento de los personajes es inmejorable y los efectos sonoros resultan sorprendentes.

Todos estos alicientes lo convierten en un juego interesante y adictivo, pese a no aportar prácticamente nada nuevo a este tipo de programas. El sonido de la presentación es aún más espectacular que el de Nodes of Yesod.

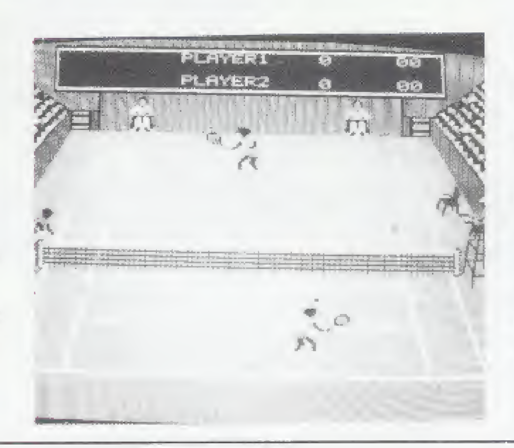
MATCH POINT

PSION

QL

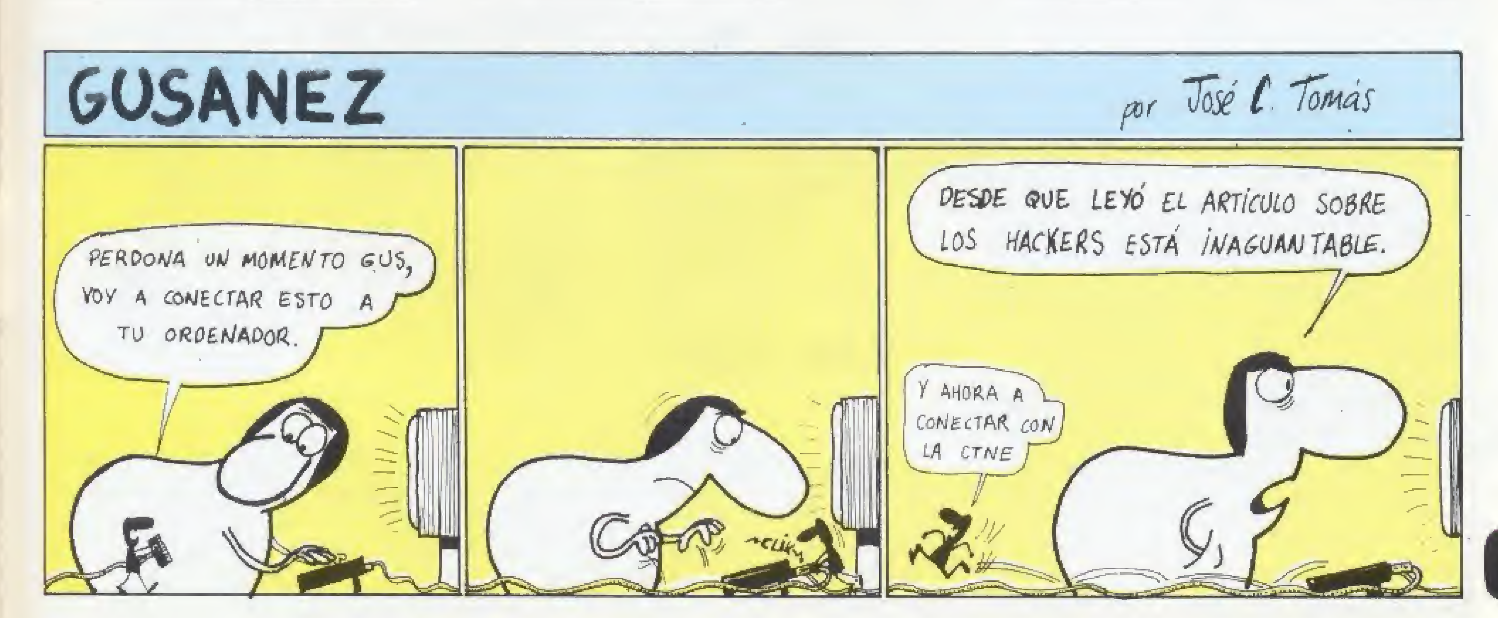


Pocas casas de *software* han dedicado sus esfuerzos a la creación de programas para el QL. Una de estas compañías, PSION, creadora del Cless y de los cuatro programas incluidos con el QL, acaba de lanzar la versión de un conocido programa de Spectrum: Match Point. Si la versión Spectrum estaba considerada como uno de los mejores programas de simulación deportiva, la versión QL la supera ampliamente. El control se realiza mediante teclado o Joystick. El juego del ordenador es muy fuerte, por lo que resulta difícil ganar una partida, incluso en el nivel más bajo. También podemos seleccionar la duración de las partidas, entre 1, 3 ó 5 sets. Las reglas del tenis han sido respetadas al máximo, por lo que este juego puede considerarse como una perfecta simulación de tenis por ordenador. Los gráficos están muy logrados, aprovechando ampliamente las posibilidades de la máquina. El sonido, aunque escaso, cumple perfectamente su cometido. El principal inconveniente es que resulta difícil acostumbrarse a la perspectiva, fallando muchos golpes por no calcular exactamente la situación de la bola. En definitiva, se trata de un juego muy adictivo que exige gran concentración y reflejos si queremos competir seriamente contra la máquina.



GUSANEZ

por José C. Tomás



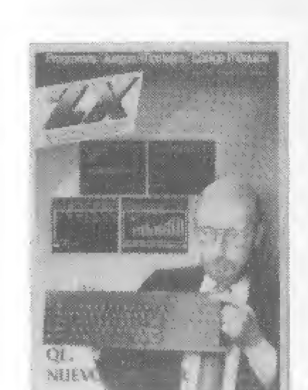


SERVICIO DE

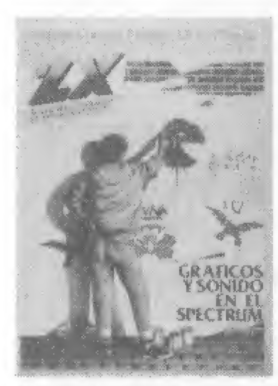
Completa tu colección de ZX.
A continuación te resumimos el contenido de los ejemplares atrasados en existencia.



Núm. 3/300 ptas.
El Spectrum por dentro. Quince programas, juegos y montajes Software.



Núm. 4/300 ptas.
QL, el nuevo Sinclair. Dieciocho programas, juegos, montajes, ideas/Novedades.



Núm. 5/300 ptas.
Gráficos y sonido en el Spectrum/Libros/Software/13 programas.



Núm. 6/300 ptas.
Construya su propio juego/13 programas y montajes/ideas/Software.



Núm. 7/300 ptas.
Juegos inteligentes/Software/ 11 programas/Libros.



Núm. 8/300 ptas.
La aventura es la aventura/12 programas/Juegos y montajes/Código máquina.



Núm. 9/300 ptas.
Construye tu propio juego. Catorce programas para el verano. Gráficos en el Spectrum.



Núm. 10/300 ptas.
Catorce programas educativos: geografía, cramer, gráficos, razones trigonométricas, elongación. Código máquina.



Núm. 11/300 ptas.
Cómo crear marcianos y otros monstruos.
Diez programas satélites de júpiter, rescate, interés, círculo, préstamo hipotecario.



Núm. 12/300 ptas.
Presentación del Spectrum Plus. Forth, capítulo 1. Gráficos en el Spectrum, 4 parte. Libros. Programas y montajes.



Núm. 13/300 ptas.
Guía del software para el Spectrum todos los programas del mercado. Forth, capítulo 2. Visitamos Sinclair Research. Libros. Programas.



Núm. 14/300 ptas.
Cómo jugar al Hobbit. Gráficos de funciones. Programas de ajedrez. Conexiones con el P I/O. Programas Multiplic, enseñar deleitando. Libros, Forth, tercera parte.



Núm. 15/300 ptas.
Simuladores de vuelo, Forth, cuarta parte. Montajes: Reloj digital para Spectrum. BASIC para principiantes. Libros. Programas.



Núm. 16/300 ptas.
Cassettes: solución a los problemas de grabación. Test de Psicología. Sistema de Desarrollo para el ZX-81. Cinemática. Programas. Animación Gráfica. BASIC para principiantes (2). Forth, quinta parte.



Núm. 17/300 ptas.
Mapa de Atic-Atac. Estira de caracteres. Dinámica de una partícula. Libros. QL Magazine. Programas. Convertidor analógico-digital con el P I/O.

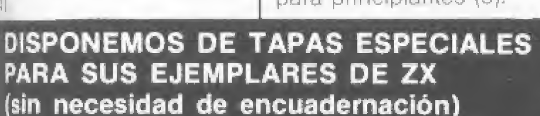
EJEMPLARES ATRASADOS



Núm. 18/300 ptas.
Rentas 85. Forth, sexta parte. Programas. BASIC para principiantes (3). Plotting Gráficos. Libros. Usuarios. Crítica.



Núm. 19/300 ptas.
Mapa de Knight Lore. Noticias. Crítica. Renta 85 (segunda parte). Libros. El ZX-81 aprende a sumar. Scroll de ventanas. Programas. El software que nos invade. BASIC para principiantes (4).



DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE ZX (sin necesidad de encuadernación)



Núm. 20/300 ptas.
Vacaciones con informática. Crítica. Noticias. Programas. Son muy divertidos. Libros. Generación de placas de circuito impreso. Forth. Movimiento armónico simple. Spectrum musical.



Núm. 21/300 ptas.
Mapa de Underwurlde. Noticias. Crítica. ¿Has probado? Programa especial: barquitos. Sois muy divertidos. Libros para el verano. Un poco de física. BASIC para principiantes (5).



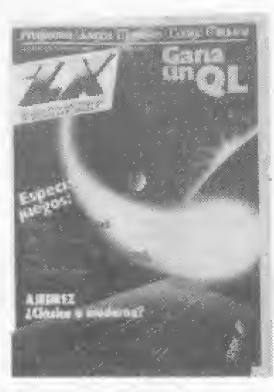
Núm. 22/300 ptas.
Noticias. Teclados profesionales. Crítica. ¿Has probado? Programa especial: procesador de textos. Generación de placas de circuito impreso (segunda parte). Programas QL español. Quinielas en Spectrum. BASIC para principiantes (6).



Núm. 23/300 ptas.
Crítica. ¿Has probado? Profanation profanado. Noticias. Discos para Spectrum. Dossier educación: Spectrum en el aula, autoevaluación, Logo. Código máquina. Programación especial: quinielas. Montaje a cámara lenta. BASIC para principiantes (7).



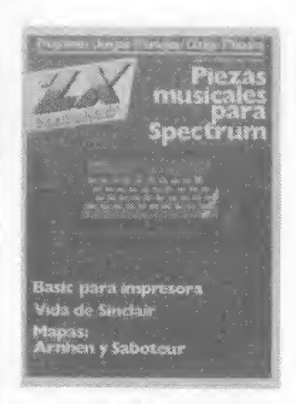
Núm. 24/300 ptas.
Juegos/Mapas del Nodes of Yesod y Lords of Midnight/¿Has probado?/Sois muy divertidos/Usuario/Ajuste de gráficas/Multisearch/Programas/Montaje: inversor de vídeo para ZX 81/Dossier QL.



Núm. 25/300 ptas.
Juegos/Especial juegos, Mapas y trucos de: Highway encounter, Tir Na Nog, Nightshade/¿Qué es el Stack?/Programa especial/Código máquina/Lotería primitiva/Stándares de la informática/Programas.



Núm. 26/300 ptas.
Spectrum o QL, invasión de los 128/¿Cómo utilizar mejor el microdrive?/Juegos/Mapa del Dun Darach y misión imposible/Programación estructurada/BASIC.



Núm. 27/300 ptas.
La vida de Sinclair/Piezas musicales para Spectrum/Juegos/Mapas del ARNHCM y SABOTEUR/Areas/BASIC para impresora/El área de variable y la instrucción RST 16.



PRECIO UNIDAD **650** ptas.

(en cada tomo se pueden encuadernar 6 números)

Para hacer tu pedido, rellena el cupón adjunto, córtalo y envíalo HOY MISMO a:

ZX, Bravo Murillo, 377 • 28020-MADRID • Tel. 733 74 13

Los ejemplares atrasados de ZX serán una fuente constante de conocimientos, ideas, soluciones y entretenimientos para el futuro. Todo lo anterior hace recomendable que los guardes ordenadamente en una de las tapas especiales para ZX. Cada tapa puede contener 6 ejemplares y cuesta solamente 650 ptas.

Ruego me envíen los siguientes ejemplares atrasados de ZX .... al precio de 300 ptas. cada uno

Por favor envíen .... tapa(s) al precio de 650 ptas. cada una (+ gastos de envío).

El importe lo abonaré:
☐ contra reembolso ☐ cheque adjunto ☐ con mi tarjeta de crédito
☐ American Expres ☐ Visa ☐ Interbank.

Fecha de caducidad ....

Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

NOMBRE ....
DIRECCION ....
POBLACION .... C.P. ....
PROVINCIA ....



Base de datos para microdrive

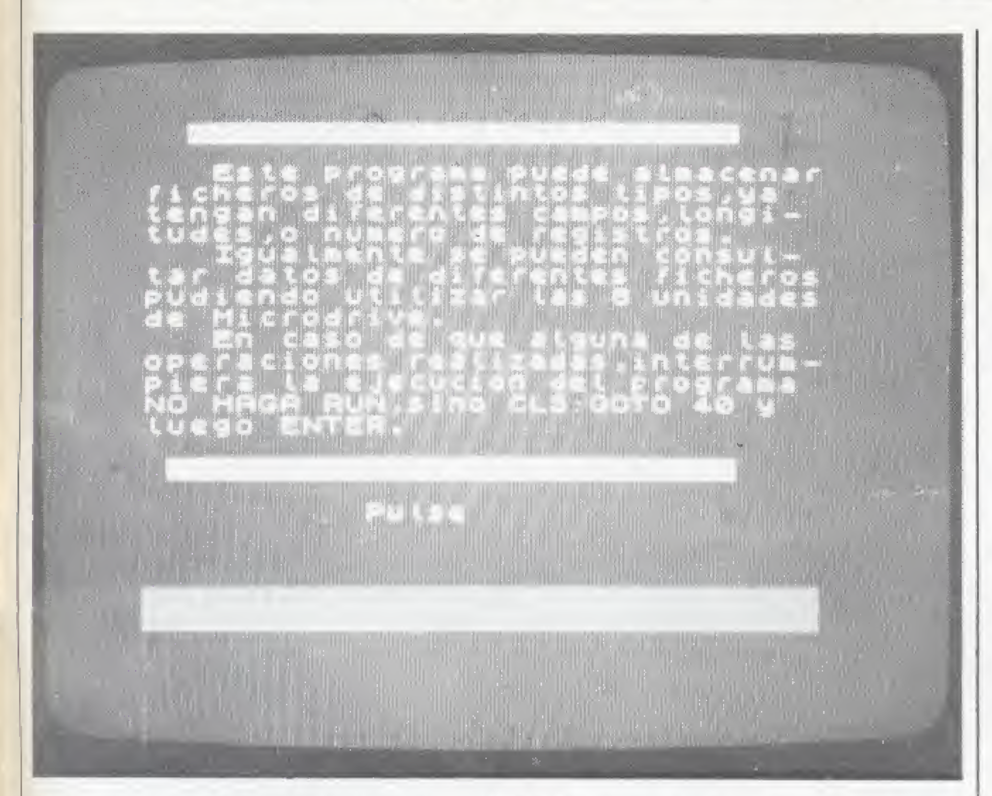
Aunque extenso, el programa que nos ha enviado desde Huelva Francisco Moreno es una completísima base de datos para microdrive. En realidad se trata de un fichero que puede manipular indistintamente otros ficheros con registros de longitud variable. Su capacidad está limitada únicamente por la del microdrive y cuantos más soportes tengamos, más ficheros podremos alcanzar.
Las opciones disponibles son las de cualquier programa semejante: formateo, intercambio de datos, altas, bajas, búsqueda de registros, clasificación, listado...

```
  10 BORDER SGN PI: PAPER SGN PI
 : INK VAL "7": CLS : LOAD *"M";1
 ;"pantalla"SCREEN$ : FOR A=60 TO
  10 STEP -1: BEEP .001,A: BORDER
  (a/10): NEXT A: PAUSE NOT PI
  20 CLEAR : LET CAT=NOT PI: LET
 CAR=NOT PI: LET REG=NOT PI: LET
 FI=VAL "1": BORDER 4: PAPER SGN
 PI: INK VAL "7": LET CRE=NOT PI
 : LET b$="": LET c$="": FOR a=NO
T PI TO VAL "31": LET b$=b$+" ":
 LET c$=c$+".": NEXT a: POKE VAL
 "23658",VAL "8": CLS
  40 IF CAT=SGN PI OR FI>SGN PI
THEN  PRINT "FICHERO<";E$;">";"
N.CAMPOS<";B;">"'"FECHA<";FE;">
HORA<";HO;">"'"REG:TOT<";REG;">"
;"LIB.<";REG-(FI-1);">";"CRE.<";
FI-1;">": GO TO VAL "60"
  50 IF NOT CAT THEN  PRINT "FIC
HERO<.........>";" N.CAMPOS<..>"
'"FECHA<......> HORA<....>"'"REG
: TOT<....>LIB<....>CRE<....>"
  60 PRINT  PAPER 4;B$
  70 PRINT "01) CREACION DE UN F
ICHERO": PRINT "02) ALTAS DE REG
ISTROS": PRINT "03) BAJAS DE REG
ISTROS": PRINT "04) MODIFICACION
 DE REGISTROS": PRINT "05) BUSQU
EDA DE REGISTROS"
  80 PRINT "06) CLASIFICACION DE
 REGISTROS": PRINT "07) LISTADO
TOTAL DE REGISTROS": PRINT "08)
LISTADO PARCIAL DE REGISTROS": P
RINT "09) CAMBIO DATOS ENTRE FIC
HEROS"
  90 PRINT "10) BORRAR DATOS DE
MEMORIA": PRINT "11) COPIA DE SE
GURIDAD": PRINT "12) FORMATEADO
DE CARTUCHOS": PRINT "13) CARGAR
 FICHERO DE DATOS": PRINT "14) G
RABAR FICHERO DE DATOS": PRINT "
15) DATOS DE UN SOLO CAMPO"
 100 PRINT "16) BORRAR UN FICHER
```



```
O DE DATOS": PRINT "17) CATALOGO
S": PRINT "18) FINALIZAR"
 120 PRINT #SGN PI;AT NOT PI,NOT
 PI;"PULSE OPCION": LET LM=VAL "
2": GO SUB VAL "400"
 130 IF CODE F$( TO 1)>57 OR COD
E F$( TO 1)<48 THEN  GO TO VAL "
120"
 140 IF LEN F$>SGN PI THEN  IF C
ODE F$(2 TO 2)>57 OR CODE F$(2 T
O 2)<48 THEN  GO TO VAL "120"
 145 IF VAL F$>NOT PI AND VAL F$
<19 THEN  GO TO VAL "190"
 150 IF VAL F$<SGN PI OR VAL F$>
VAL "18" THEN  GO TO VAL "120"
 190 GO TO 500+500*VAL F$
 400 REM SUBR.ESCRITURA PALABRA
 405 BEEP .01,60: PRINT #SGN PI;
AT NOT PI,VAL "25";"(*)MENU": PR
INT #SGN PI;AT SGN PI,SGN PI;C$(
 TO LM);">";B$( TO 32-2-LM);AT S
GN PI,NOT PI;"<":
 410 LET F$="": FOR X=1 TO LM: P
AUSE NOT PI: LET G$=INKEY$
 420 IF G$=CHR$ 13 THEN  GO TO V
AL "470"
 430 IF G$=CHR$ 12 AND LEN F$>NO
T PI THEN  PRINT #SGN PI;CHR$ 8;
".";CHR$ 8;: BEEP .01,0: PAUSE N
OT PI: LET G$=INKEY$: LET F$=F$(
 TO LEN F$-1): LET X=X-1: GO TO
VAL "420"
 440 IF CODE G$<32 OR CODE G$>12
7 THEN  LET X=X-1: GO TO 460
 450 PRINT #SGN PI;G$;: BEEP .01
,40: LET F$=F$+G$: IF F$( TO 1)=
"*" THEN  CLS : INPUT "": GO TO
VAL "40"
 460 NEXT X
 470 PRINT #SGN PI;AT NOT PI,VAL
 "23"; FLASH SGN PI;"CORRECTO?":
 PAUSE NOT PI: LET A$=INKEY$: IF
 A$="n" OR a$="N" THEN  PRINT #S
GN PI;AT NOT PI,VAL "23";B$( TO
9);AT SGN PI,SGN PI;C$( TO LM);A
T SGN PI,NOT PI; OVER SGN PI;" "
;: LET F$="": LET X=NOT PI: GO T
O VAL "460"
 480 IF A$="s" OR A$="S" THEN  P
RINT #SGN PI;AT NOT PI,VAL "23";
B$( TO 9): GO TO VAL "500"
 490 GO TO VAL "470"
 500 INPUT "": RETURN
 510 STOP
1000 REM CREACION DE UN FICHERO
1005 IF cre THEN  PRINT #SGN PI;
AT NOT PI,NOT PI;"FICHERO EN MEM
ORIA. SI CONTINUO SERA BORRADO."
: BEEP .01,10: PAUSE VAL "200":
INPUT "": PRINT #SGN PI;AT NOT P
I,NOT PI;"CONTINUO? (SI/NO)": PA
USE NOT PI: LET a$=INKEY$: IF a$
="s" OR a$="S" THEN  GO TO VAL "
20"
1006 IF cre AND a$="n" OR a$="N"
 THEN  GO TO VAL "120"
1010 CLS : PRINT  INVERSE 1;"CRE
ACION"
1025 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NUMERO DE CAMPOS": LET LM=V
AL "2": GO SUB VAL "400": LET B=
VAL F$
1030 IF B>VAL "15" THEN  PRINT #
SGN PI;AT SGN PI,NOT PI; FLASH S
GN PI;"15 CAMPOS COMO MAXIMO": P
AUSE VAL "200": GO TO VAL "1010"
1035 IF B<SGN PI THEN  PRINT #SG
N PI;AT SGN PI,NOT PI; FLASH SGN
 PI;"1 CAMPO COMO MINIMO": PAUSE
 VAL "200": GO TO VAL "1010"
1040 DIM d$(B,20): LET f$="": FO
R A=1 TO B: PRINT #SGN PI;AT NOT
 PI,NOT PI;"NOMBRE DEL CAMPO ";A
;": ": LET LM=20: GO SUB 400
1099 LET D$(A)=F$: PRINT F$
1100 NEXT A: LET cre=SGN PI: CLS
1110 PRINT TAB 3; INVERSE 1;"CAM
POS";TAB 20; INVERSE 1;"LONGITUD
."
1120 FOR A=1 TO B: PRINT AT 3+A,
INT PI;d$(a);"<";c$( TO 2);">":
NEXT A
1125 DIM d(b): PRINT #SGN PI;AT
NOT PI,NOT PI;"LONGITUD :MAX.30-
MIN.1"
1130 FOR c=1 TO b
1135 PRINT AT 3+c,24;".";CHR$ 8;
1140 LET LM=VAL "2": GO SUB VAL
"400": IF VAL F$>30 OR VAL F$<SG
N PI THEN  GO TO VAL "1140"
1150 PRINT F$: LET D(C)=VAL F$:
NEXT C: GO TO VAL "1210"
1210 LET totme=NOT PI: FOR a=1 T
O b: LET totme=totme+d(a): NEXT
a: LET reg=INT (((65535-USR 7962
)-2000)/totme): PRINT #SGN PI;AT
 SGN PI,NOT PI;"FICHERO CON ";re
g;" REGISTROS"
1215 PAUSE VAL "200": INPUT "":
PRINT #SGN PI;AT NOT PI,NOT PI;"
FECHA DE HOY": LET LM=VAL "6": G
O SUB VAL "400": LET FE=VAL F$:
INPUT "": LET LM=VAL "4": PRINT
#SGN PI;AT NOT PI,NOT PI;"HORA A
CTUAL": GO SUB VAL "400": LET HO
=VAL F$
1220 FOR a=1 TO b: GO SUB 1230+a
: NEXT a: IF CAR=1 THEN  LET CAR
=NOT PI: GO TO 7060
1230 GO TO VAL "1250"
1231 DIM H$(reg+1,d(a)): RETURN
1232 DIM I$(reg,d(a)): RETURN
1233 DIM J$(reg,d(a)): RETURN
1234 DIM K$(reg,d(a)): RETURN
1235 DIM L$(reg,d(a)): RETURN
1236 DIM M$(reg,d(a)): RETURN
1237 DIM N$(reg,d(a)): RETURN
1238 DIM O$(reg,d(a)): RETURN
1239 DIM P$(reg,d(a)): RETURN
```

```
1240 DIM Q$(reg,d(a)): RETURN
1241 DIM R$(reg,d(a)): RETURN
1242 DIM S$(reg,d(a)): RETURN
1243 DIM T$(reg,d(a)): RETURN
1244 DIM U$(reg,d(a)): RETURN
1245 DIM V$(reg,d(a)): RETURN
1255 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NOMBRE DEL FICHERO": BEEP .
01,40: LET LM=VAL "10": GO SUB V
AL "400": LET E$=F$: PRINT #SGN
PI;AT NOT PI,NOT PI;"FICHERO CRE
ADO": BEEP .01,30: PAUSE VAL "10
0": CLS : GO TO VAL "9800"
1500 REM ALTAS DE REGISTROS
1510 IF CRE=NOT PI THEN  PRINT #
SGN PI;AT NOT PI,NOT PI;"NO EXIS
TE FICHERO EN MEMORIA"'B$: BEEP
.01,40: PAUSE VAL "100": INPUT "
": GO TO VAL "120"
1550 CLS : GO TO VAL "9800"
2000 REM BAJAS DE REGISTROS
2010 IF NOT CRE OR FI<2 THEN  PR
INT #SGN PI;AT NOT PI,NOT PI;"NO
 EXISTEN FICHAS EN MEMORIA."'B$:
 BEEP .01,30: PAUSE VAL "100": I
NPUT "": GO TO VAL "120"
2020 CLS : PRINT  INVERSE SGN PI
;"BAJAS DE REGISTROS": PRINT ''"
   Para dar de baja un registro,
debera introducir el numero del
mismo"
2030 PRINT #SGN PI;AT NOT PI,NOT
 PI;"N. DEL REGISTRO": LET LM=VA
L "4": GO SUB VAL "400": LET H=V
AL F$
2035 IF h>fi-1 THEN  PRINT #SGN
PI;AT NOT PI,NOT PI;"NO EXISTE E
SE REGISTRO": BEEP .1,0: PAUSE V
AL "100": GO TO VAL "2030"
2040 FOR C=1 TO B: FOR A=H TO FI
-1: GO SUB VAL "2100+C": NEXT A:
 NEXT C: LET FI=FI-1: PRINT #SGN
 PI;AT NOT PI,NOT PI;"REGISTRO B
ORRADO": PAUSE VAL "50": CLS : G
O TO VAL "40"
2101 LET H$(A)=H$(A+1): RETURN
2102 LET I$(A)=I$(A+1): RETURN
2103 LET J$(A)=J$(A+1): RETURN
2104 LET K$(A)=K$(A+1): RETURN
2105 LET L$(A)=L$(A+1): RETURN
2106 LET M$(A)=M$(A+1): RETURN
2107 LET N$(A)=N$(A+1): RETURN
2108 LET O$(A)=O$(A+1): RETURN
2109 LET P$(A)=P$(A+1): RETURN
2110 LET Q$(A)=Q$(A+1): RETURN
2111 LET R$(A)=R$(A+1): RETURN
2112 LET S$(A)=S$(A+1): RETURN
2113 LET T$(A)=T$(A+1): RETURN
2114 LET U$(A)=U$(A+1): RETURN
2115 LET V$(A)=V$(A+1): RETURN
2400 PAUSE 0
2500 REM MODIFICACION REGISTROS
2510 IF NOT CRE THEN  PRINT #SGN
 PI;AT NOT PI,NOT PI;"NO EXISTE
FICHERO EN MEMORIA"'B$: PAUSE VA
L "100": INPUT "": GO TO VAL "12
0"
3000 REM BUSQUEDA DE REGISTROS
3010 IF NOT CRE THEN  PRINT #SGN
 PI;AT NOT PI,NOT PI;"NO HAY NIN
GUN FICHERO EN MEMORIA"'B$: PAUS
E VAL "100": INPUT "": GO TO VAL
 "120"
3020 CLS : PRINT  INVERSE SGN PI
;"BUSQUEDA DE UN REGISTRO": PRIN
T '': FOR A=SGN PI TO B: PRINT A
;") ";D$(A): BEEP .01,60: NEXT A
3030 PRINT #SGN PI;AT NOT PI,NOT
 PI;"PULSE CAMPO A BUSCAR": LET
LM=2: GO SUB VAL "400"
3040 LET G=VAL F$
3050 PRINT #SGN PI;AT NOT PI,NOT
 PI;"DATO A BUSCAR";B$( TO 8): L
ET LM=D(G): GO SUB VAL "400": LE
T W$=F$
3060 FOR A=SGN PI TO FI-SGN PI:
GO SUB VAL "3095"
3070 NEXT A: PRINT #SGN PI;AT NO
T PI,NOT PI;"NO EXISTEN MAS REGI
STROS": BEEP .01,30: PAUSE VAL "
100": PRINT #SGN PI;AT NOT PI,NO
T PI;"DESEA BUSCAR OTRO REGISTRO
?": BEEP .01,20: PAUSE NOT PI: L
ET A$=INKEY$: IF A$="S" OR A$="s
" THEN  GO TO VAL "3020"
3080 CLS : GO TO VAL "40"
3095 FOR E=1 TO LEN W$: GO SUB V
AL "3100"+G*2
3102 IF H$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3103 RETURN
3104 IF I$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3105 RETURN
3106 IF J$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3107 RETURN
3108 IF K$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3109 RETURN
3110 IF L$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3111 RETURN
3112 IF M$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3113 RETURN
3114 IF N$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3115 RETURN
3116 IF O$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3117 RETURN
3118 IF P$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3119 RETURN
3120 IF Q$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3121 RETURN
3122 IF R$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3123 RETURN
3124 IF S$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3125 RETURN
3126 IF T$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3127 RETURN
3128 IF U$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3129 RETURN
3130 IF V$(A)( TO E)=W$( TO E) T
HEN  NEXT E: GO SUB VAL "3200":
RETURN
3131 RETURN
3200 CLS : PRINT  INVERSE SGN PI
;"REGISTRO ENCONTRADO ";A: FOR C
=1 TO B: PRINT : PRINT C;")";D$(
C)'" <";: GO SUB VAL "3300"+C: P
RINT ">": NEXT C
3210 PRINT '''#SGN PI;AT NOT PI,
NOT PI;"ES EL REGISTRO BUSCADO?"
: PAUSE NOT PI: LET F$=INKEY$
3220 IF f$="S" OR f$="s" THEN  P
RINT #SGN PI;AT NOT PI,NOT PI;"B
USCA OTRO REGISTRO?";B$( TO 4):
PAUSE NOT PI: LET f$=INKEY$: GO
TO VAL "3240"
3225 IF F$<>"S" AND F$<>"s" THEN
  GO TO VAL "3070"
3240 IF f$="s" OR f$="S" THEN  G
O TO VAL "3020"
3250 CLS : GO TO VAL "40"
3301 PRINT H$(A);: RETURN
3302 PRINT I$(A);: RETURN
3303 PRINT J$(A);: RETURN
3304 PRINT K$(A);: RETURN
3305 PRINT L$(A);: RETURN
3306 PRINT M$(A);: RETURN
3307 PRINT N$(A);: RETURN
3308 PRINT O$(A);: RETURN
3309 PRINT P$(A);: RETURN
3310 PRINT Q$(A);: RETURN
3311 PRINT R$(A);: RETURN
3312 PRINT S$(A);: RETURN
3313 PRINT T$(A);: RETURN
3314 PRINT U$(A);: RETURN
3315 PRINT V$(A);: RETURN
3500 REM CLASIFICACION REGISTROS
3510 IF NOT CRE THEN  PRINT #SGN
 PI;AT NOT PI,NOT PI;"NO EXISTE
FICHERO EN MEMORIA"'B$: PAUSE VA
L "100": GO TO VAL "120"
3520 CLS : PRINT  INVERSE SGN PI
;"CLASIFICACION DE REGISTROS": P
RINT : FOR A=1 TO B: PRINT A;")"
;D$(A): NEXT A
3530 PRINT #SGN PI;AT NOT PI,NOT
 PI;"QUE CAMPO CLASIFICO?": LET
LM=VAL "2": GO SUB VAL "400"
3580 FOR A=1 TO FI-1
3700 STOP
4000 REM LISTADO TOTAL REGISTROS
4010 IF NOT CRE OR FI<2 THEN  PR
INT #SGN PI;AT NOT PI,NOT PI;"NO
 EXISTEN FICHAS EN MEMORIA."'B$:
 BEEP .01,35: PAUSE VAL "100": I
NPUT "": GO TO VAL "120"
4015 LET L1=SGN PI: LET L2=FI-SG
N PI
4020 CLS : FOR A=L1 TO L2: CLS :
 PRINT TAB 10; INVERSE SGN PI;"R
EGISTRO N.";A
4025 FOR C=1 TO B: PRINT 'C;")";
D$(C)
4030 PRINT "<";: BEEP .01,45: GO
 SUB C+4100: PRINT ">"
4040 NEXT C: PAUSE 0
4050 NEXT A: PRINT '''': PRINT #
SGN PI;AT NOT PI,NOT PI;"FIN DE
FICHERO"'"PULSE UNA TECLA PARA C
ONTINUAR": BEEP .1,10: PAUSE NOT
 PI: CLS : GO TO VAL "40"
4101 PRINT H$(A);: RETURN
4102 PRINT I$(A);: RETURN
4103 PRINT J$(A);: RETURN
4104 PRINT K$(A);: RETURN
4105 PRINT L$(A);: RETURN
4106 PRINT M$(A);: RETURN
4107 PRINT N$(A);: RETURN
4108 PRINT O$(A);: RETURN
4109 PRINT P$(A);: RETURN
4110 PRINT Q$(A);: RETURN
4111 PRINT R$(A);: RETURN
4112 PRINT S$(A);: RETURN
4113 PRINT T$(A);: RETURN
4114 PRINT U$(A);: RETURN
4115 PRINT V$(A);: RETURN
4500 REM LIST. PARCIAL REGISTROS
4510 IF NOT CRE THEN  PRINT #SGN
 PI;AT NOT PI,NOT PI;"NO EXISTE
FICHERO EN MEMORIA": PAUSE VAL "
100": INPUT "": GO TO VAL "120"
4520 CLS : PRINT  INVERSE SGN PI
;"LISTADO PARCIAL DE REGISTROS"
4530 PRINT #SGN PI;AT NOT PI,NOT
 PI;"DESDE QUE N.DE REGISTRO?":
LET LM=VAL "4": GO SUB VAL "400"
: LET L1=VAL F$: IF L1<SGN PI TH
EN  GO TO VAL "4520"
4540 PRINT #SGN PI;AT NOT PI,NOT
 PI;"HASTA QUE N.DE REGISTRO?":
LET LM=VAL "4": GO SUB VAL "400"
: LET L2=VAL F$: IF L2>FI-SGN PI
 OR L2<L1 THEN  INPUT "": GO TO
VAL "4540"
4550 GO TO VAL "4020"
5000 REM DATOS ENTRE FICHEROS
```




```
5010 CLS : PRINT  INVERSE SGN PI
;"DATOS ENTRE FICHEROS"
5020 PRINT '"1)COPIO FICHERO EN
OTRO MICRODR."''"2)TODOS LOS DAT
OS POR IMPRESORA"''"3)CAMBIO EL
NOMBRE A UN FICHERO"''"4)TODOS L
OS DATOS POR PANTALLA"
5030 PRINT #SGN PI;AT NOT PI,NOT
 PI;"PULSE OPCION": LET LM=SGN P
I: GO SUB VAL "400": GO TO 5000+
(VAL F$*100)
5100 PRINT #SGN PI;AT NOT PI,NOT
 PI;"N.MICRODRIVE FUENTE": LET L
M=SGN PI: GO SUB VAL "400": LET
FU=VAL F$
5110 PRINT #SGN PI;AT NOT PI,NOT
 PI;"N.MICRODRIVE DESTINO": LET
LM=SGN PI: GO SUB VAL "400": LET
 DE=VAL F$
5120 MOVE "M";FU;E$ TO "M";DE;E$
: PRINT SGN PI;AT NOT PI,NOT PI;
"PULSE UNA TECLA PARA CONTINUAR"
: PAUSE NOT PI: CLS : GO TO VAL
"40"
5200 PRINT #SGN PI;AT NOT PI,NOT
 PI;"N. DE MICRODRIVE": LET LM=S
GN PI: GO SUB VAL "400": LET G=V
AL F$: CLS : MOVE "M";G;E$ TO #3
: PRINT #SGN PI;AT NOT PI,NOT PI
;"PULSE UNA TECLA PARA CONTINUAR
": PAUSE NOT PI: CLS : GO TO VAL
 "40"
5300 CLS : PRINT  INVERSE SGN PI
;"CAMBIAR NOMBRE A UN FICHERO":
PRINT ''"   Para cambiar el nomb
re a un  fichero, debe estar gra
bado en  el cartucho."
5302 PRINT #SGN PI;AT NOT PI,NOT
 PI;"N.DE MICRODRIVE": LET LM=SG
N PI: GO SUB VAL "400": INPUT ""
: PRINT ' INVERSE SGN PI;"Ficher
os del microdrive n.";VAL f$: CA
T VAL f$
5310 PRINT  INVERSE SGN PI'"Fich
ero en memoria <";E$;">"
5320 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NOMBRE ANTIGUO": LET LM=VAL
 "10": GO SUB VAL "400": LET W$=
F$: PRINT #SGN PI;AT NOT PI,NOT
PI;"NOMBRE NUEVO";B$( TO 5): LET
 LM=VAL "10": GO SUB VAL "400":
LET E$=F$: PRINT #SGN PI;AT NOT
PI,NOT PI;"N. DEL MICRODRIVE": L
ET LM=SGN PI: GO SUB VAL "400":
LET F=VAL F$: ERASE "M";F;W$: CL
S
5330 GO TO VAL "7535"
5400 CLS : PRINT  INVERSE SGN PI
;"OBTENCION DE DATOS POR PANTALL
A": PRINT #SGN PI;AT NOT PI,NOT
PI;"N. DE MICRODRIVE": LET LM=SG
N PI: GO SUB VAL "400": LET F=VA
L F$: PRINT ' INVERSE SGN PI;"Fi
cheros del microdrive n.";f: CAT
 F: INPUT "": PRINT #SGN PI;AT N
OT PI,NOT PI;"DE QUE FICHERO?":
LET LM=VAL "10": GO SUB VAL "400
": INPUT "": MOVE "M";F;F$ TO #2
5410 PRINT #SGN PI;AT NOT PI,NOT
 PI;"PULSE UNA TECLA": PAUSE NOT
 PI: CLS : GO TO VAL "40"
5500 REM BORRAR DATOS MEMORIA
5510 RUN 20
6000 REM COPIA DE SEGURIDAD
6010 IF CRE=SGN PI THEN  PRINT #
SGN PI;AT NOT PI,NOT PI;"DEJE LI
BRE LA MEMORIA . SI QUIE-RE, SAL
VE ANTES EL FICH.DE DATOS": PAUS
E VAL "300": INPUT "": GO TO VAL
 "120"
6012 CLS : PRINT  INVERSE SGN PI
;"COPIA DE SEGURIDAD": PRINT ''"
1) COPIA DE PROGRAMA Y PANTALLA.
2) COPIA DE PROGRAMA.": LET LM=S
GN PI: GO SUB VAL "400": IF VAL
F$=SGN PI THEN  INPUT "": GO TO
VAL "6020"
6014 IF VAL F$=2 THEN  PRINT #SG
N PI;AT NOT PI,NOT PI;"NOMBRE DE
 LA COPIA?": LET LM=VAL "10": GO
 SUB VAL "400": LET W$=F$: INPUT
 "": PRINT #SGN PI;AT NOT PI,NOT
 PI;"N.MICRODRIVE?": LET LM=SGN
PI: GO SUB VAL "400": INPUT "":
PRINT #SGN PI;AT NOT PI,NOT PI;"
GRABANDO <";W$;">": SAVE *"M";VA
L F$;W$ LINE 1: PRINT #SGN PI;AT
 NOT PI,NOT PI;"VERIFICANDO <";W
$;">": VERIFY *"M";VAL F$;W$: GO
 TO VAL "20"
6016 GO TO VAL "6012"
6020 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NOMBRE DE LA COPIA?": LET L
M=VAL "10": GO SUB VAL "400": LE
T W$=F$: PRINT #SGN PI;AT NOT PI
,NOT PI;"N. DE MICRODRIVE";B$( T
O 4): LET LM=SGN PI: GO SUB VAL
"400": PRINT #SGN PI;AT NOT PI,N
OT PI;"INTRODUZCA EL CARTUCHO CO
N EL   PROGRAMA ORIGINAL Y PULSE
 TECLA.": PAUSE NOT PI: LOAD *"M
";VAL F$;"pantalla"SCREEN$ : PRI
NT #SGN PI;AT NOT PI,NOT PI;"CAM
BIE DE CARTUCHO PARA LA COPIADE
SEGURIDAD Y PULSE UNA TECLA.": P
AUSE NOT PI: INPUT "": SAVE *"m"
;VAL f$;"pantalla"SCREEN$ : PRIN
T #SGN PI;AT NOT PI,NOT PI;"GRAB
ANDO PROGRAMA <";W$;">": SAVE *"
M";VAL F$;W$ LINE 1: CLS : GO TO
 VAL "40"
6500 REM FORMATEADO DE CARTUCHOS
6510 CLS : PRINT  INVERSE SGN PI
;"FORMATEADO DE CARTUCHOS": PRIN
T #SGN PI;AT NOT PI,NOT PI;"NOMB
RE DEL CARTUCHO": LET LM=VAL "10
": GO SUB VAL "400": LET E$=F$:
PRINT #SGN PI;AT NOT PI,NOT PI;"
NUMERO DEL MICRODRIVE": LET LM=S
GN PI: GO SUB VAL "400": LET A=V
AL F$: PRINT #SGN PI;AT NOT PI,N
OT PI;"FORMATEANDO ";"<";E$;">";
B$( TO 7): FORMAT "M";A;E$: CLS
: GO TO VAL "40"
7000 REM CARGAR FICHERO DE DATOS
7003 IF CRE=SGN PI THEN  PRINT #
SGN PI;AT NOT PI,NOT PI;"HAY FIC
HERO EN MEMORIA. DEBE U- SAR ANT
ES LA OPCION 10": BEEP .01,30: P
AUSE VAL "200": GO TO VAL "120"
7005 CLS : PRINT  INVERSE SGN PI
;"CARGANDO FICHEROS"
7010 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NUMERO DE MICRODRIVE": LET
LM=SGN PI: GO SUB VAL "400": LET
 F=VAL F$: CAT F: PRINT #SGN PI;
AT NOT PI,NOT PI;"FICHERO A CARG
AR": LET LM=VAL "10": GO SUB VAL
 "400"
7020 LET CAT=SGN PI: LET CAR=SGN
 PI: LET A$=F$: PRINT #SGN PI;AT
 NOT PI,NOT PI;"CARGANDO <";a$;"
>"
7030 OPEN #4;"M";F;A$
7040 INPUT #4;FE: INPUT #4;HO: I
NPUT #4;CRE: INPUT #4;REG: INPUT
 #4;FI: INPUT #4;B: INPUT #4;E$
7044 DIM D(B): DIM D$(B,20)
7045 FOR A=1 TO B: INPUT #4;D$(A
): INPUT #4;D(A): NEXT A
7050 GO TO 1220
7060 FOR A=SGN PI TO FI-1
7070 GO SUB B+7100: NEXT A: CLOS
E #4: LET CAR=NOT PI: CLS : GO T
O VAL "40"
7101 INPUT #4;H$(A): RETURN
7102 INPUT #4;H$(A);I$(A): RETUR
N
7103 INPUT #4;H$(A);I$(A);J$(A):
 RETURN
7104 INPUT #4;H$(A);I$(A);J$(A);
K$(A): RETURN
7105 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A): RETURN
7106 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A): RETURN
7107 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A): RETURN
7108 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);;N$(A);O$(A):
RETURN
7109 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A): RETURN
7110 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A): RETURN
7111 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A);R$(A): RETURN
```

```
7112 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A);R$(A);S$(A): RETURN
7113 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A);R$(A);S$(A);T$(A): RET
URN
7114 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A);R$(A);S$(A);T$(A);U$(A
): RETURN
7115 INPUT #4;H$(A);I$(A);J$(A);
K$(A);L$(A);M$(A);N$(A);O$(A);P$
(A);Q$(A);R$(A);S$(A);T$(A);U$(A
);V$(A): RETURN
7500 REM GRABACION DEL FICHERO
7505 IF CRE=NOT PI THEN  PRINT #
SGN PI;AT NOT PI,NOT PI;"NO HAY
FICHERO EN MEMORIA": PAUSE VAL "
100": INPUT "": GO TO VAL "120"
7510 PRINT #SGN PI;AT NOT PI,NOT
 PI;"UNIDAD DE MICRODRIVE": LET
LM=SGN PI: GO SUB VAL "400": LET
 F=VAL F$: CLS : CAT F: PRINT ''
'"<";E$;"> ES EL NOMBRE DEL FICH
ERO QUE ESTA EN MEMORIA."
7515 PRINT #SGN PI;AT NOT PI,NOT
 PI;"BORRA FICHERO CON IGUAL NOM
BRE  DEL CATALOGO?(S/N)": BEEP .
1,25: PAUSE NOT PI: LET F$=INKEY
$: IF F$="S" OR F$="s" THEN  PRI
NT #SGN PI;AT NOT PI,NOT PI;"BOR
RANDO <";E$;">";B$( TO 10);'B$:
ERASE "m";f;e$: GO TO VAL "7530"
7520 IF F$="N" OR F$="n" THEN  G
O TO VAL "7530"
7525 GO TO VAL "7515"
7530 INPUT "": PRINT #SGN PI;AT
NOT PI,NOT PI;"FECHA DE HOY": LE
T LM=VAL "6": GO SUB VAL "400":
LET FE=VAL F$: INPUT "": LET LM=
```



```
VAL "4": PRINT #SGN PI;AT NOT PI
,NOT PI;"HORA ACTUAL": GO SUB VA
L "400": LET HO=VAL F$
7535 PRINT #SGN PI;AT NOT PI,NOT
 PI;"GRABANDO <";E$;">"'B$: BEEP
 .01,60: OPEN #4;"m";F;E$
7540 PRINT #4;FE: PRINT #4;HO: P
RINT #4;CRE: PRINT #4;REG: PRINT
 #4;FI: PRINT #4;B: PRINT #4;E$:
 FOR A=1 TO B: PRINT #4;D$(A): P
RINT #4;D(A): NEXT A
7550 FOR A=1 TO FI-1
7560 GO SUB B+7600: NEXT A: LET
H$(FI)="*": PRINT #4;H$(FI): CLO
SE #4: PRINT #SGN PI;AT NOT PI,N
OT PI;"GRABADO  <";E$;">": BEEP
.01,30: PAUSE VAL "100": CLS : G
O TO VAL "40"
```

```
7601 PRINT #4;H$(A): RETURN
7602 PRINT #4;H$(A)'I$(A): RETUR
N
7603 PRINT #4;H$(A)'I$(A)'J$(A):
 RETURN
7604 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A): RETURN
7605 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A): RETURN
7606 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A): RETURN
7607 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A): RETURN
7608 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A): R
ETURN
7609 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A): RETURN
7610 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A): RETURN
7611 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A)'R$(A): RETURN
7612 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A)'R$(A)'S$(A): RETURN
7613 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A)'R$(A)'S$(A)'T$(A): RET
URN
7614 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A)'R$(A)'S$(A)'T$(A)'U$(A
): RETURN
7615 PRINT #4;H$(A)'I$(A)'J$(A)'
K$(A)'L$(A)'M$(A)'N$(A)'O$(A)'P$
(A)'Q$(A)'R$(A)'S$(A)'T$(A)'U$(A
)'V$(A): RETURN
8000 REM DATOS DE UN SOLO CAMPO
```

```
8010 IF NOT CRE THEN  PRINT #SGN
 PI;AT NOT PI,NOT PI;"NO EXISTE
FICHERO EN MEMORIA"'B$: BEEP .01
,50: PAUSE VAL "100": INPUT "":
GO TO VAL "120"
8020 CLS : PRINT  INVERSE SGN PI
;"DATOS DE UN SOLO CAMPO"
8030 PRINT : FOR A=1 TO B: PRINT
 A;")";D$(A): BEEP .01,50: NEXT
A: PRINT #SGN PI;AT NOT PI,NOT P
I;"PULSE CAMPO": LET LM=VAL "2":
 GO SUB VAL "400": LET H=VAL F$
8040 CLS : PRINT  INVERSE SGN PI
;AT NOT PI,NOT PI;"CAMPO:<";D$(H
);">": FOR A=SGN PI TO FI-SGN PI
: PRINT : GO SUB VAL "3300+H": B
EEP .01,50: NEXT A
8050 PRINT #SGN PI;AT NOT PI,NOT
 PI;"DESEA DATOS DE OTRO CAMPO?"
: BEEP .01,20: PAUSE NOT PI: LET
 G$=INKEY$: IF G$="S" OR G$="s"
THEN  GO TO VAL "8020"
8060 CLS : GO TO VAL "40"
8500 REM BORRAR FICHERO DE DATOS
8510 CLS : PRINT  INVERSE SGN PI
;"BORRAR UN FICHERO DE DATOS": P
RINT : PRINT #SGN PI;AT NOT PI,N
OT PI;"N.MICRODRIVE?": LET LM=SG
N PI: GO SUB VAL "400": LET E=VA
L F$: PRINT  INVERSE SGN PI;"CAT
ALOGO DEL MICRODRIVE N. ";E: CAT
 E: PRINT #SGN PI;AT NOT PI,NOT
PI;"FICHERO A BORRAR?": LET LM=V
AL "10": GO SUB VAL "400": INPUT
 "": PRINT #SGN PI;AT NOT PI,NOT
 PI;"BORRANDO <";f$;">": ERASE "
M";E;F$
8520 PRINT #SGN PI;AT NOT PI,NOT
 PI;B$;AT SGN PI,NOT PI;"FICHERO
 <";F$;"> BORRADO": BEEP .01,35:
 PAUSE VAL "100"
8530 PRINT #SGN PI;AT NOT PI,NOT
 PI;"BORRA OTRO FICHERO?": LET L
M=SGN PI: GO SUB VAL "400": IF F
$="S" OR F$="s" THEN  GO TO VAL
"8500"
8540 CLS : GO TO VAL "40"
9000 REM CATALOGOS
9005 POKE 23692,0
9010 PRINT #SGN PI;AT NOT PI,NOT
 PI;"NUMERO DEL MICRODRIVE": LET
 LM=SGN PI: GO SUB VAL "400"
9020 CLS : PRINT  INVERSE SGN PI
;"CATALOGO DEL MICRODRIVE N. ";V
AL F$: CAT VAL F$
9030 PRINT #SGN PI;AT NOT PI,NOT
 PI;"OTRO CATALOGO?";B$( TO 10):
 BEEP .01,55: PAUSE NOT PI: LET
a$=INKEY$: IF a$="s" OR a$="S" T
HEN  GO TO VAL "9010"
9040 IF a$="n" OR a$="N" THEN  C
LS : GO TO VAL "40"
9050 GO TO VAL "9030"
9500 REM FINALIZAR
9510 PRINT USR NOT PI
9800 REM SUBR. ALTAS REGISTROS
9805 FOR e=fi TO reg: CLS : PRIN
T TAB VAL "10"; INVERSE SGN PI;"
REGISTRO N. ";FI
9810 FOR a=SGN PI TO B: PRINT d$
(a): PRINT #SGN PI;AT NOT PI,NOT
 PI;D$(A): POKE 23692,255: LET L
M=D(A): GO SUB 400: PRINT #SGN P
I;AT NOT PI,NOT PI;B$;B$: PRINT
"<";F$;">": PRINT #SGN PI;AT NOT
 PI,NOT PI;B$
9901 IF A=SGN PI THEN  LET H$(E)
=F$: GO TO VAL "9920"
9902 IF A=2 THEN  LET I$(E)=F$:
GO TO VAL "9920"
9903 IF A=3 THEN  LET J$(E)=F$:
GO TO VAL "9920"
9904 IF A=4 THEN  LET K$(E)=F$:
GO TO VAL "9920"
9905 IF A=5 THEN  LET L$(E)=F$:
GO TO VAL "9920"
9906 IF A=6 THEN  LET M$(E)=F$:
GO TO VAL "9920"
9907 IF A=7 THEN  LET N$(E)=F$:
GO TO VAL "9920"
9908 IF A=8 THEN  LET O$(E)=F$:
GO TO VAL "9920"
9909 IF A=9 THEN  LET P$(E)=F$:
GO TO VAL "9920"
9910 IF A=10 THEN  LET Q$(E)=F$:
 GO TO VAL "9920"
9911 IF A=11 THEN  LET R$(E)=F$:
 GO TO VAL "9920"
9912 IF A=12 THEN  LET S$(E)=F$:
 GO TO VAL "9920"
9913 IF A=13 THEN  LET T$(E)=F$:
 GO TO VAL "9920"
9914 IF A=14 THEN  LET U$(E)=F$:
 GO TO VAL "9920"
9915 IF A=15 THEN  LET V$(E)=F$:
 GO TO VAL "9920"
9920 LET F$="": NEXT A
9930 LET FI=FI+SGN PI: PAUSE VAL
 "50": NEXT e
9940 PRINT #SGN PI;AT NOT PI,NOT
 PI;"FICHERO <";E$;"> COMPLETO":
 PAUSE VAL "200": CLS : GO TO VA
L "40"
9997 REM GRABACION
9998 STOP
9999 CLS : ERASE "M";1;"run": ER
ASE "m";1;"run1": SAVE *"m";1;"r
un" LINE 10: SAVE *"m";1;"run1"
LINE 1: VERIFY *"m";1;"run": PRI
NT "CORRECTAMENTE GRABADO <run>"
: VERIFY *"m";1;"run1": PRINT '"
CORRECTAMENTE GRABADO <run1>": P
RINT '''': CAT 1
```



APARTADO DE CORREOS

Dirige tus cartas a:
Todospectrum
Bravo Murillo, 377, 5.º-A
28020 Madrid

EL REGISTRO INDICE IY

Al tratar de estudiar algunas SUBRUTINAS DE LA ROM, encuentro, con frecuencia, que recurren al CONTENIDO DEL REGISTRO INDEXADO «IY», sin que se haya cargado antes en el curso de la subrutina. ¿A qué es ello debido? Desearía saber si existe algún libro donde vengan explicadas las SUBRUTINAS DE LA ROM, aunque no esté en castellano.

José M.ª Contreras
Osuna (Sevilla)

El registro IY es usado por el sistema operativo del Spectrum para indexar las variables del sistema, es por ello por lo que siempre apunta a la dirección 23610 (ERR NR). Por lo tanto, IY + 14 equivaldrá a la variable del sistema BORDCR (23624) y IY − 50 será LAST K (23560). No es conveniente utilizar el registro IY en nuestras rutinas si no hemos desactivado las interrupciones con DI, pues esto haría que las rutinas que incrementan FRAMES y exploran el teclado no funcionaran correctamente, además de que podrían modificar las posiciones que estamos indexando con IY. Si utilizamos este sistema, es necesario permitir las interrupciones (EI) y devolver a IY su valor correspondiente (23610) antes de retornar al BASIC. Aunque, de hecho, la rutina a la que se salta (STACK_BC) realiza esta operación antes que nada, podría ocurrir que se produjera una interrupción antes de que esto se ejecutara.

El libro que interesa no es otro que el famoso *The Complete Spectrum ROM Disassembly* de Ian Logan y Frank O'Hara, publicado por la editorial inglesa Melburne House Publishers. En él podrás encontrar el desensamblado de la totalidad de la ROM del Spectrum con comentarios (en inglés) de cada uno de los pasos.

MAS SOBRE LOS REGISTROS IX E IY

¿Podemos (en código máquina) manejar directamente los registros IX e IY?. Y si no, ¿para qué sirven? ¿Cómo se usan las instrucciones BIT, CCF, DI, IND, LDD, NEG, NOP, RES y SET?

Javier Burgués
Lérida

Puedes utilizar los registros IX e IY con toda comodidad en tus programas en ensamblador, admitirán todas las instrucciones que sueles utilizar con el par HL, pero cuando direccionen alguna posición de memoria, deberán ir seguidos de un byte en complemento a dos. Por ejemplo, puedes usar perfectamente LD A, (IX+3). Has de tener más cuidado con el par IY, pues es utilizado por el operativo para indexar las variables del sistema.

En cuanto a la «ristra» de instrucciones sobre las que pides consejo, es evidente que no es ésta la sección adecuada para tan extenso tema. Te recomendamos que sigas la serie «Aprendiendo lenguaje máquina» en la que si no han sido ya vistas esas instrucciones lo serán pronto.

ERROR EN «CINCO HORAS CON SCREEN$»

No podría dejar de felicitaros por la extraordinaria labor divulgativa que lleváis a cabo y mis deseos son de que cada día más lectores se den cuenta de ello y que no caigáis en las redes del consumismo, ya que el Spectrum, a pesar de los años que tiene y mientras no llegue la generación de los 16 bits, seguirá siendo un ordenador de talla.

Respecto al artículo «Cinco horas con SCREEN$», publicado en el número de diciembre, pienso que los duendes de la imprenta han vuelto a hacer de las suyas, pues los principiantes como yo, seguramente habrán perdido más de cinco horas en averiguar por qué tras modificar el programa de la figura 3, según instrucciones, no se consigue el efecto deseado, a saber, lograr un OVER 1.

En mi opinión debería decir: «La simulación de un OVER 1 se llevará a cabo mediante la función XOR, cambiaremos la línea 8 por XOR (HL)».

Por otro lado, el artículo «Un nuevo operativo para el Spectrum» me ha parecido muy bueno. Enhorabuena, Manuel Arana, me gustaría mantener contacto directo por correspondencia contigo. Mis señas son:

Juan J. Jiménez
Mapocho, 7
29018 Málaga

Efectivamente existe la pequeña errata que comentas, ese 6 que se «coló» en lugar del 8 que correspondía. Aunque no creemos que sea tan grave como para hacer perder más de esas cinco horas que prometía el artículo, debemos pedir disculpas tanto a ti como a quienes resultaran despistados por este error.

COPY CON EL INTERFACE 1

Tengo ante mí el número 12 de su revista, donde, en la sección de Preguntas y Respuestas, se ofrece el listado de una rutina de ensamblador para hacer COPY de pantalla con una impresora ADMATE DP-100. En el texto que acompaña a la rutina se señala que está preparada para ser utilizada con un interface tipo Kempston, y se comentan las modificaciones necesarias para utilizarla con el Interface 1 y su salida RS-232. Pues bien, o yo no he comprendido las modificaciones que debo hacer o alguna otra cosa no funciona, pues al ejecutar la rutina mi impresora imprime cualquier cosa menos lo que hay en pantalla en ese momento. ¿Es posible que esto sea debido a las variaciones entre las distintas versiones de Interface 1?, el mío es un ISSUE 2.

Luis C. Durán
Madrid

El error que detectas no tiene nada que ver con las diferencias entre versiones del Interface 1 sino más bien con un despiste nuestro cuando mencionamos las modificaciones necesarias. A diferencias de los interfaces tipo Kempston, el Interface 1 sólo necesita recibir el caracter de escape (CHR$27) antes del primer código de control. Por ello no habia que eliminar las líneas de impresión de los escapes cuando se dieran dos veces seguidas a lo largo del programa sino que había que eliminarlos siempre excepto cuando se repetían, en cuyo caso debían aparecer una sola vez.

```
 10 COPY
 20             LD   A,3
 30             CALL #1601
 40             LD   A,27
 50             RST  16
 60             LD   A,"3"
 70             RST  16
 80             LD   A,23
 90             RST  16
100             LD   HL,#4000
110 NULI
120             LD   A,13
130             RST  16
140             LD   A,10
150             RST  16
160             LD   A,H
170             CP   #58
180             RET  Z
190             LD   A,27
200             RST  16
210             LD   A,"K"
220             RST  16
230             LD   A,0
240             RST  16
250             LD   A,1
260             RST  16
270 NOLI
280             LD   B,8
290 NUCA
300             PUSH BC
310             LD   B,8
320             LD   DE,TABLA
330 NUBY
340             RLC  (HL)
350             EX   DE,HL
360             RL   (HL)
370             INC  HL
380             EX   DE,HL
390             DJNZ NUBY
400             INC  H
410             POP  BC
420             DJNZ NUCA
430             EX   DE,HL
440             LD   B,8
450             LD   HL,TABLA
460 IMPR
470             LD   A,(HL)
480             RST  16
490             INC  HL
500             DJNZ IMPR
510             EX   DE,HL
520             LD   C,L
530             INC  L
540             JR   Z,NULI
550             LD   A,H
560             SUB  8
570             LD   H,A
580             LD   A,C
590             AND  #1F
600             CP   #1F
610             JR   NZ,NOLI
620             JR   NULI
630 TABLA
640             DEFS 8
```

CAMPAÑA DIFAMATORIA

La carta especial de este mes nos la remite desde Sevilla José Guzmán, quien frecuentemente se pone en contacto con nosotros para intercambiar información y opiniones sobre el QL.

Estimados amigos de QL Magazine y Todospectrum:

Os escribo porque me ha llegado un nuevo rumor dentro de la campaña difamatoria contra el QL. Lo último es que se puede autodestruir por software.

Te lo presentan diciendo que como tiene dos procesadores, se les puede liar para que se ataquen y se destruyan. Dado que la única forma posible de dañarlo sería activar dos chips sobre el mismo bus a la vez, y eso viene activado por hardware, es físicamente imposible dañar cualquier ordenador por software.

Este bulo es una auténtica canallada contra el QL. ¿Qué les ha hecho el QL para que lo odien tanto? Creo que ya está bien. Los usuarios estamos hartos de que el QL sea el muñequito del pim-pam-pum.

Se oculta la existencia de máquinas derivadas del QL como el One Per Desk de ICL, el Merlín Tonto de British Telecom y el Computerphone de Australia Telecom, compañías telefónicas de Inglaterra y Australia respectivamente. Son versiones modificadas del QL que incorporan teclado, modem, teléfono y dos microdrives. A diferencia del QL, llevan los cuatro programas de Psion en ROM y el SuperBASIC en microdrive.

Esto vuelve a demostrar la increíble «competencia» técnica de los críticos del QL; creo que ICL y las dos telefónicas les han dado la mejor respuesta.

Saludos.

José M. Guzmán
Sevilla

EL CORCHO

¡Ganga! Vendo Atari 800 XL, cassette Atari, 3 juegos, 2 joysticks, impresora Atari 1027 y Atariwriter, todo por 60.000 ptas. También lo cambiaría todo por impresora matricial Seikosha o similar. Pilar de Rivas. Urb. Levantina «La Solana». Tarragona. Tel.: (977) 23 74 32.

Compro, vendo, cambio revistas Ordenador Popular, Todospectrum y ZX. Baratísimas. Alberto. Tel.: (981) 26 34 41 de 6 a 8 excepto lunes y miércoles. (La Coruña).

Vendo Amstrad CPC 664, monitor color, unidad de disco 3" (en garantía), programas de origen más software valorado en 20.000 ptas., joystick Quickshot Ii y manual firmware. Aceptaré la mejor oferta a partir de 85.000 ptas. C. García de Castro. Riera Alta, 43, 2.º, 1-A. 08001 Barcelona.

Vendo QL 128 K con cuatro programas (hoja de cálculo, tratamiento de textos, control de stocks, gráficos) más uno de juegos y varios libros. Javier. Tel.: (93) 751 23 49. Barcelona.

Vendo ZX-Spectrum 48 K por cambio a 128 K. 20.000 ptas. Perfecto estado, pocas horas de uso. Regalo paquete de programas a elección del interesado. Preguntar por Luis. Tel.: (91) 243 80 57.

Compro o cambio compilador de lenguaje C par QL. Si lo tienes escribe a Miguel Echevarría Martínez. Avda. General Yagüe, 20. 09004 Burgos o llama al Tel.: (947) 21 96 71.

CATALOGO DE SOFTWARE PARA ORDENADORES PERSONALES IBM

TODO EL CATALOGO DE SOFTWARE CON MAS DE 800 FICHAS



OFERTA ESPECIAL DE SUSCRIPCION

1.ª ENTREGA 3.500,— PTAS.
(400 FICHAS + FICHERO)

RESTO EN TRES ENTREGAS TRIMESTRALES DE 1.500,— PTAS. CADA UNA.

PRECIO TOTAL DE LA SUSCRIPCION - 8.000,— PTAS.

CUPON DE PEDIDO

SOLICITE **HOY MISMO** EL CATALOGO DIRECTAMENTE A

infodis, s.a.

BRAVO MURILLO, 377 - 5.° A
28020 MADRID

O EN LOS CONCESIONARIOS IBM

El importe lo abonaré: POR CHEQUE ☐ CONTRA REEMBOLSO ☐
CON MI TARJETA DE CREDITO ☐ Ref: CATALOGO DE SOFTWARE

Cargue **8.000** ptas. a mi tarjeta American Express ☐ Visa ☐ Interbank

Número de mi tarjeta ____________________

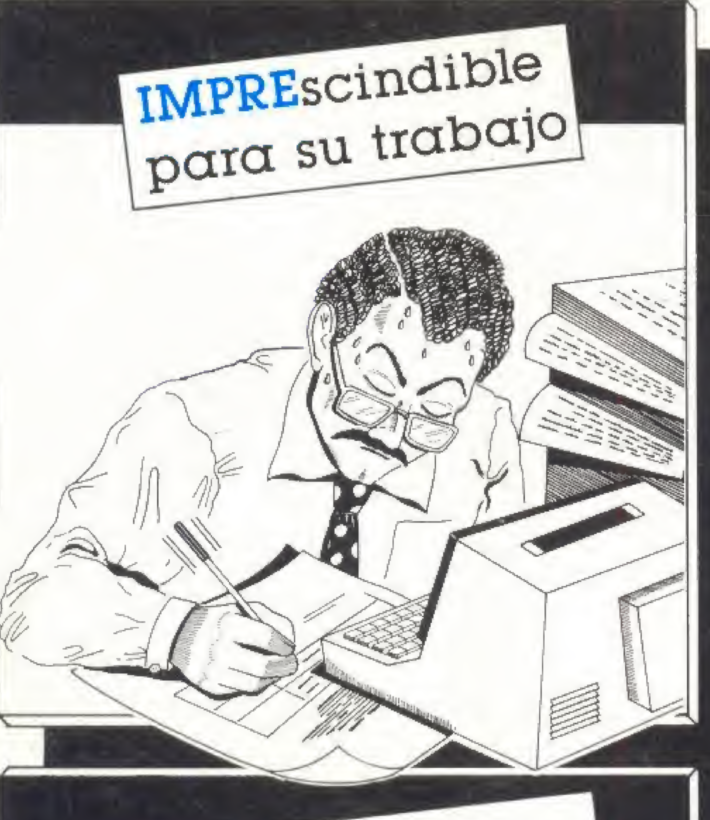
Fecha de caducidad ____________ Firma

NOMBRE ____________________

CALLE ____________________

CIUDAD ____________ D.P. ________

PROVINCIA ____________________

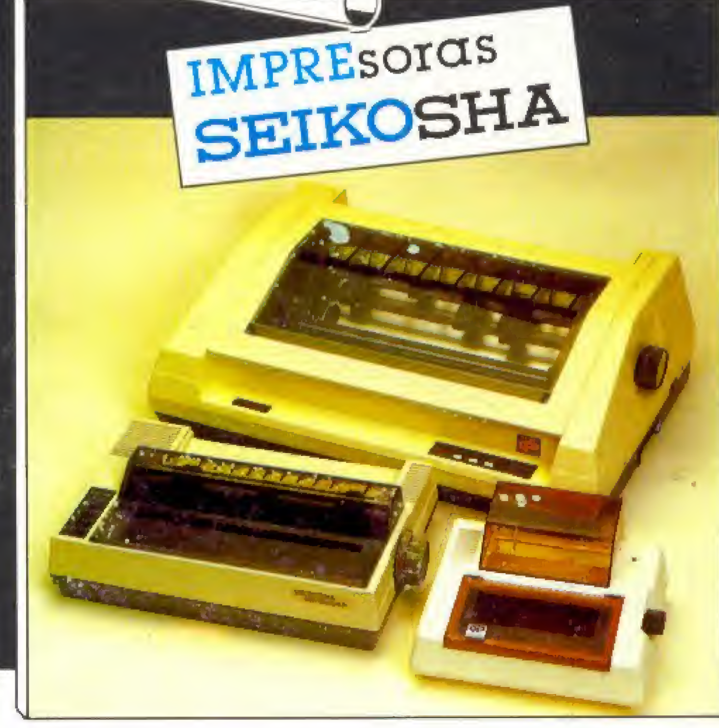












GP-50 •	La pequeña 40 cps. Papel normal con interface paralelo, serial y Spectrum	17.990 ptas.
GP-700 *	La de color 50 cps. 7 colores. 80 columnas. Tracción y fricción. Papel de 10 pulgadas	64.990 ptas.
SP-1.000 *	La programable 100 cps.24 cps en alta calidad 96 cart. programables en RAM. Introductor hoja a hoja.♦	64.990 ptas.
SP-1.000AS	La programable 100 cps.24 cps en alta calidad con interface RS-232. Introductor hoja a hoja.♦	59.900 ptas.
MP-1.300AI	La polivalente 300 cps, 60 cps en alta calidad, interface paralelo y RS-232. Introductor hoja a hoja.♦&	119.900 ptas.
BP-5.200 *	La de oficina 200 cps, 106 en alta calidad.Buffer 4K.Carro de 15".Tracción y fricción.♦	199.900 ptas.
BP-5.420 *	La más rápida 420 cps. 106 cps en alta calidad. Buffer de 18K. Paralelo y RS-232.♦	339.900 ptas.

Interfaces: Serie RS-232C, Spectrum, IBM, COMMODORE, MSX, QL, Apple Macintosh, HP-IB

♦ Introductor automático de documentos opcional.

& Kit de color opcional.

* con interface paralelo

• con interface Spectrum

Nota: I.V.A. 12%, no incluido en los precios arriba indicados

Avda. Blasco Ibáñez, 116
Tel. (96) 372. 88. 89
Telex 62220 - 46022 VALENCIA

Muntaner, 60-2.º-4.ª
Tel. (93) 323. 32. 19
08011 BARCELONA

Agustin de Foxá, 25-3.º-A
Tels. (91) 733. 57. 00 - 733. 56. 50
28036 MADRID

